Master's Thesis : LISP Mapping System Under Attack

Auteur : Gabriel, Mattias
Promoteur(s) : Donnet, Benoît
Faculté : Faculté des Sciences appliquées
Diplôme : Master : ingénieur civil en informatique, à finalité spécialisée en "computer systems security"
Année académique : 2019-2020
URI/URL : https://gitlab.com/m.gabriel/lisp-ms-ddos; http://hdl.handle.net/2268.2/9059

UNIVERSITY OF LIÈGE
FACULTY OF APPLIED SCIENCES

MASTER THESIS

LISP Mapping System under attack

Author:
Mattias GABRIEL

Supervisor:
Pr. Benoît DONNET

Jury:
Pr. Guy LEDUC
Pr. Bernard BOIGELOT

Graduation Studies conducted for obtaining the Master's degree in Master of Science in Computer Science and Engineering by Mattias Gabriel

Academic year 2019 – 2020

Abstract

The Locator/Identifier Separation Protocol (LISP) is an encapsulation protocol currently in development. It is based on the potential need to reorganize the routing architecture of the Internet in order to meet the still increasing size of this worldwide network. The key principle of this protocol is to split the current IP address space into an identifier address space and a locator one. In this paradigm, the identifier address serves the purpose of identifying a connection endpoint and is only routable in a stub network, a LISP site. The locator address, in turn, is used to locate this site in the core network. This address is thus globally routable. For nodes from different LISP sites to communicate between each other, a data tunnel has to be put in place between both sites.

Because of this separation principle, LISP needs a mechanism allowing it to translate an address from the identifier space to the locator space: the mapping system. Thanks to this, a LISP site is able to query a mapping, binding both address spaces, by the use of LISP control messages. LISP-DDT is a notable example of mapping system which draws inspiration, regarding its architecture, from the Domain Name System.

Both LISP and LISP-DDT current implementations may be prone to potential security vulnerabilities. In this regard, this work aims at getting a clear understanding of the security aspects of the studied protocols. This approach is done in order to find potential vulnerabilities in these protocols – while not claiming to be exhaustive – and take advantage of them in order to develop an attack. That way, a denial-of-service attack by amplification has been found out. This attack exploits the mapping lookup process between a LISP site and the mapping system. In particular, it relies on the fact that the mapping system is able to generate responses that are significantly larger than the queries causing them. This principle can hence be used to produce a lot of network traffic towards a predetermined victim node in order to consume its bandwidth.

As a proof-of-concept for the attack, a GNS3 emulated network topology has been set up and configured. This network therefore simulates an up and running LISP-DDT mapping system – mimicking the one of the LISP Beta Network, a worldwide deployment of LISP on Internet – in order to use it as an amplification vector for the attack. Results of the attack on this enclosed environment are analysed in this work. It proves the feasibility of the attack in the current implementations of LISP and LISP-DDT.

Finally, a brief discussion about possible mitigation techniques for the attack is provided. Among these mitigation techniques, one can cite the limitation of the reply size, the rate limitation or even anti-spoofing techniques. Either way, we hope to draw the LISP IETF Working Group's attention to the necessity of addressing this issue.

Acknowledgements

First of all, I would like to express my gratitude to Benoit Donnet, my thesis supervisor, for his advices and encouragements throughout the year. He was always available for helping me to carry out and organise this Master thesis. For giving me strong interest in computer science and security, I am thankful.

Along with Benoit Donnet, I would also like to thank Luigi Iannone for giving me helpful advices and support regarding LISP. I really appreciated working and conversing with him about this topic.

This work would not have been possible without the help of Cyril Soldani and Justin Iurman. Both of them kindly allowed me to operate resources and materials from the Montefiore Institute. Moreover, they have generously given of their time in order to help me set up the devices needed for the experiments.

Last but not least, I wish to express my heartfelt thanks to my friends and family for providing me moral supports all along the development of this work. I would really like to mention Antoine, Archibald, Arnaud, Bertrand, Cédric, Melissa and of course Noémie for their precious help. Finally, a special thank also goes to Véronique.

Contents

List of Figures

List of Tables

List of Abbreviations

AFI	**A**ddress **F**amily **I**dentifier
BGP	**B**order **G**ateway **P**rotocol
DFZ	**D**efault-free **z**one
DHT	**D**istributed **H**ash **T**able
DNS	**D**omain **N**ame **S**ystem
DTLS	**D**atagram **T**ransport **L**ayer **S**ecurity
ECM	**E**ncapsulated **C**ontrol **M**essage
EID	**E**ndpoint **ID**entifier
ETR	**E**gress **T**unnel **R**outer
IH	**I**nner **H**eader
ITR	**I**ngress **T**unnel **R**outer
ISP	**I**nternet **S**ervice **P**rovider
KDF	**K**ey **D**erivation **F**unction
LCAF	**L**ISP **C**anonical **A**ddress **F**ormat
LISP	**L**ocator/**I**dentifier **S**eparation **P**rotocol
LISP-DDT	**LISP D**elegated **D**atabase **T**ree
MAC	**M**essage **A**uthentication **C**ode
NAT	**N**etwork **A**ddress **T**ranslation
OH	**O**uter **H**eader
OTK	**O**ne-**t**ime **k**ey
PETR	**P**roxy **E**gress **T**unnel **R**outer
PITR	**P**roxy **I**ngress **T**unnel **R**outer
RLOC	**R**outing**LOC**ator
TCP	**T**ransmission **C**ontrol **P**rotocol
TLD	**T**op-**L**evel **D**omain
UDP	**U**ser **D**atagram **P**rotocol
VPN	**V**irtual **P**rivate **N**etwork
xTR	**ITR** or **ETR**

Chapter 1

Introduction

Since its beginning, the Internet is in continuous expansion and it still keeps growing nowadays [3]. Thereupon, solutions may have to be taken in order to provide the needed scalability of this network [4]. In particular, the ongoing growth of the routing tables in the core Internet may be a concern. A reorganization of the Internet routing architecture may thus become a necessity. Notably, the IP address semantics as it is today is overloaded : it serves at the same time an identification and a localisation purpose. Currently, such an address is in one hand used to identify a specific host in the network but is in the other hand used for figuring out on which site this host is connected. One of the possible solutions to adopt for this reorganization is to split the IP address into two roles: the locator and the identifier.

The Locator/Identifier Separation Protocol (LISP) addresses this issue by defining two address types, and thus splitting the IP address into two roles: the identifier and the locator [5]. First, the *identifier* is used to identify a node in the network. Such an address, referred as an Endpoint Identifier (EID), is only locally routable, in a specific LISP site. The second address type, the *locator* serves the purpose of locating a LISP site in the Internet. This second address, called the Routing Locator (RLOC), is in turn globally routable. For making two nodes from different LISP sites able to communicate between each other, a data tunnel has to be put in place between these two LISP sites. This tunneling is done by encapsulating an IP packet containing EID addresses into a new IP packet which in turn contains RLOC addresses. The latter packet is therefore able to cross the core network and be routed to the destination LISP site.

Of course, a mechanism has to exist in order to translate an address from the identifier addressing space to the locator addressing space. Indeed, this mechanism is needed by the different LISP sites in order to perform the needed encapsulations. A *mapping system* is an architecture that serves the purpose of translating an address from the identifier space to the locator space [6]. In this way, a LISP site has to possibility to request relevant mappings prior to the encapsulation of packets intended for another LISP site. This request is done by mean of two main LISP control messages : the `Encapsulated Map-Request` and the `Map-Reply` messages. LISP is designed in a way that the mapping system implementation is independent from the specification of LISP. Thanks to this modularity, multiple mapping system implementations can exist, each with different architectures. LISP-DDT is a notable example of mapping system that has an architecture similar to the one of the Domain Name System (DNS) [7].

Like any other system in computer science, LISP and its mapping system may present security vulnerabilities. This is particularly the case for LISP and LISP-DDT

as they both are fairly new and currently in active development. Moreover, these protocols are intended to be deployed on the public Internet, which cannot be considered as a trusted environment. However, security aspects already exist in LISP and its mapping system. Amongst these aspects, one can site the fact that a LISP site must be authenticated with the mapping system in order to register an EID address subspace the site is responsible for. In addition, LISP-SEC is an extension of the LISP protocol that provides security features to it regarding authentication and integrity of the LISP control messages [8].

This work presents a denial-of-service attack that exploits security vulnerabilities in LISP and LISP-DDT. Those vulnerabilities allow IP spoofing techniques as well as traffic amplification to happen. Indeed, the amplification relies on the fact that the LISP-DDT mapping system is susceptible to generate `Map-Reply` messages that are significantly larger than the `Encapsulated Map-Request` messages triggering them. The main idea of the attack presented in this work is therefore to send a large amount of `Encapsulated Map-Request` messages to the mapping system while impersonating a predetermined victim. In return, the mapping system will generate an even larger traffic towards the victim and consume a significant part of its bandwidth. The exhaustion of the victim's bandwidth can lead to a denial-of-service if the severity of the attack is sufficiently high.

In order to prove the feasibility of this denial-of-service attack by amplification exploiting the LISP-DDT mapping system, a GNS3 emulated network topology has been set up. This topology includes a set of routers that are used to implement LISP as well as LISP-DDT. For the latter, the mapping system topology mimics the one of the LISP Beta Network, the wide-scale deployment of LISP on the public Internet. This network also comprises an attacker node and a victim one. The attack has been run and it experimentally proved that such an attack is indeed feasible for the current implementation of the LISP and LISP-DDT protocols. As a matter of fact, the mapping system indeed amplified the traffic generated by the attack, which consumed victim's resources such as its network bandwidth and CPU load. In order to prevent this attack to happen in the future, some mitigation techniques are also suggested in this work. In a nutshell, such mitigation can be done by disallowing spoofing in the network, by rate-limiting the traffic, by detecting denial-of-service attacks or even by preventing the mapping system to send large messages if they are not authenticated.

Chapter 2 details the functioning of LISP and its mapping system whereas Chapter 3 introduces the security aspects of these technologies. Moreover, it highlights possible security vulnerabilities in their implementation. Chapter 4 defines a denial-of-service attack and set up a testbed on which to perform it. This chapter also suggests mitigation methods to adopt in order to prevent this attack to happen in the future. Finally, Chapter 5 concludes this work.

Chapter 2

Locator/Identifier Separation Protocol

This chapter presents the Locator/Identifier Separation Protocol (LISP). The first section provides a broad overview of the protocol as well as the rationale and advantages of deploying such a protocol in today's Internet. In particular, this first section presents LISP by splitting it into its two main parts: the data plane and the control plane. A stronger emphasis is placed on the control plane as it will be used in Chapter 4 to design an attack.

The second section, in turn, presents the different mapping system designs that can be used to implement the control plane. Amongst these designs, the implementation of LISP-DDT in particular will be detailed. As a matter of fact, LISP-DDT is the mapping system implementation that is used for the remaining of the work.

2.1 LISP Introduction

2.1.1 Principle

Nowadays, the scalability of the routing system in the Default-free zone (DFZ) is more and more worrisome [4]. Indeed, even by considering the IPv4 address space exhaustion [9], the routing tables size in the core Internet keeps growing [3]. This growth is thus due to a deaggregation of the IPv4 address space [10]. This deaggregation arises from different factors such as, for instance, multi-homing, traffic engineering and the demand of provider-independent address spaces by end-sites [11, 4]. This scalability concern results, among others, in an increase of memory and processing power needed in the routers as well as a burst of `BGP UPDATE` messages crossing the DFZ.

More generally, a root cause has been highlighted : the overloading of IP Address Semantics [4]. At the present time, the IP address is both used for identification and localisation purposes. The identifier is obviously needed to identify an end-host on the network whereas the locator serves a routing purpose, in order to know where the host is located on the network. The locator addressing follows the Internet topology so as to aggregate the addresses and thus, efficiently store and advertise them. However, the identifier typically depends on an organizational structure instead of a topological one : this address does not have to depend on the location of the node. As a consequence, a single address space serving both purposes is difficult to manage and it can raise problems.

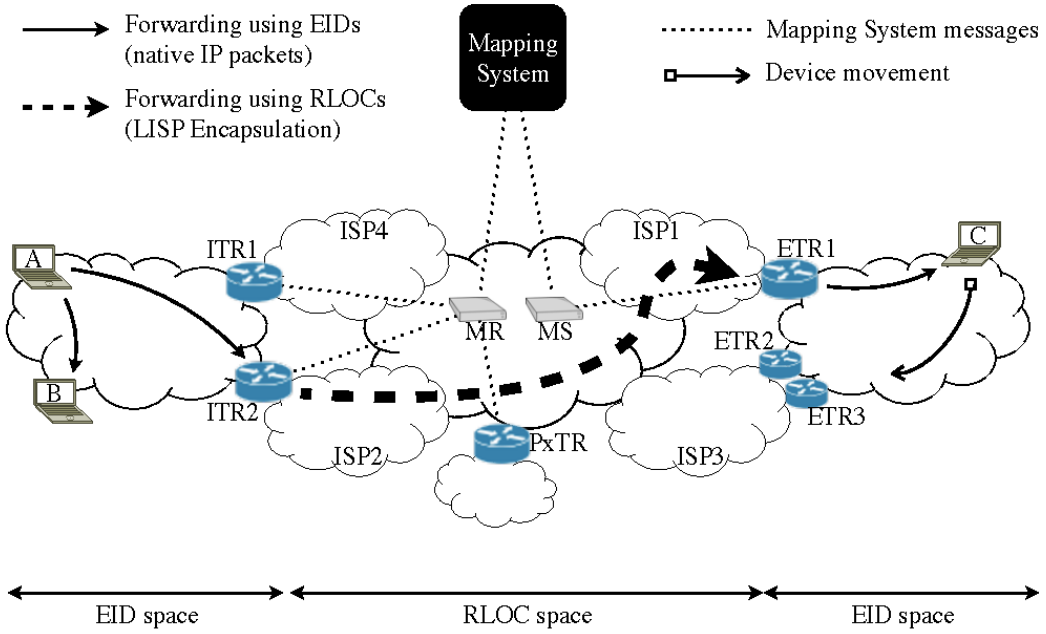

FIGURE 2.1: Routing examples from one LISP node to another [1].

The Locator/Identifier Separation Protocol [5, 12] addresses this issue by splitting the locator and the identifier space. In that respect, two namespaces coexist : the Endpoint Identifier (EID) that serves the purpose of identifying a particular node and the Routing Locator (RLOC) which is needed by the underlying routing system. As a consequence of this separation, a site is able, among others, to manage its EID topology more loosely and to change its internet provider without having to either readdress their nodes or use provider-independent address space in the first place. It is important to note that both namespaces have the same syntax as IPv4 or IPv6 addresses even though the LISP protocol allows, by design, an application to use any Address Family Identifier [13] (AFI) published by the IANA. The IPv4 and IPv6 addresses are examples of defined AFIs as well as the LISP Canonical Address Format [14] (LCAF) which extends the possible syntaxes allowed to LISP-specific encodings, such as Geo-Coordinates for instance.

The EID only has meaning on the stub networks whereas the RLOC is specific to the core network (the DFZ). An IP packet leaving an endpoint has as a source address the EID of this endpoint and as a destination address the EID of the destination host. If the destination host is located in the same domain as the source host, the packet will be forwarded in the same way as for traditional IP packets. This is the case of a packet sent by node A to node B in Figure 2.1.

However, if the destination host is located in another domain (such as the node C in Figure 2.1), the packet will first be forwarded to an Ingress Tunnel Router (ITR) – a LISP-capable router at the boundary between the source domain and the core

network. This ITR will encapsulate the packet in a new IP packet whose source address is a source RLOC and destination address is a destination RLOC mapped to the destination EID. As the RLOC addresses have a meaning in the core network, the encapsulated packet will be routed to the Egress Tunnel Router (ETR) – a LISP-capable router at the boundary between the core network and the destination domain – advertising the destination RLOC address. This ETR will decapsulate the packet and forward it towards the destination domain. The destination address of the inner packet is an EID associated to a host in the destination domain ; the packet can thus be routed to this host.

In this scenario, one can notice that the xTRs (ITRs and ETRs) are the only devices in the network that need to understand LISP. Indeed, LISP is transparent to the end hosts and the other routers ; they work in the same way as for traditional IP packets. This property benefits the LISP deployment and maintenance in a network.

In addition, a translation from the EID space to the RLOC space occurred at the ITR. Therefore, the ITRs must keep track of the different EID-to-RLOCs mappings. This is achieved by requesting the mappings in the LISP mapping system [6] – a database storing and keeping track of the different mappings – and by caching those for performance reasons [5, 15]. LISP defines a set of message types in order to communicate with the mapping system. In particular, `Map-Request` and `Map-Reply` messages are respectively used to request a mapping and send the answer back. Additionally, the `Map-Register` message is needed to register mapping updates to the mapping system. Section 2.1.3 details the different messages used to communicate with the mapping system.

The mapping system is the main part of the LISP control plane [6], which is independent from the LISP data plane. Thanks to such a design, it gives a greater modularity to LISP and it allows the data and control planes, among others, to scale differently. Due to this modularity, several mapping systems exist and have been proposed, each with a different architecture [2]. Depending on the mapping system used, the control plane might offer a better management than the underlying routing system regarding traffic engineering, multi-homing and mobility [5]. At the present time, only two mapping systems amongst the variety of existing designs have genuinely been deployed : LISP+ALT [16] – a mapping system relying on BGP [17] and tunneling – and LISP-DDT [7] – the successor of LISP+ALT that has an architecture similar to the DNS [18]. This topic is discussed in more detail in Section 2.2.

Finally, the implementation of LISP on today's internet must be progressive and the protocol has to interwork with Non-LISP sites [19]. In order to do so, two new network elements are introduced : the Proxy Ingress Tunnel Router (PITR) and Proxy Egress Tunnel Router (PETR). The former is used as an ITR for hosts in a Non-LISP site, by acknowledging the reachability of the EID prefixes in the core network whereas the latter is used by hosts in a LISP site trying to reach a host in a Non-LISP site while still using LISP encapsulation in the data plane.

2.1.2 Data plane

As previously stated, the IP packet travelling through the core network must not contain EIDs as source or destination addresses. Indeed, the EID address space has

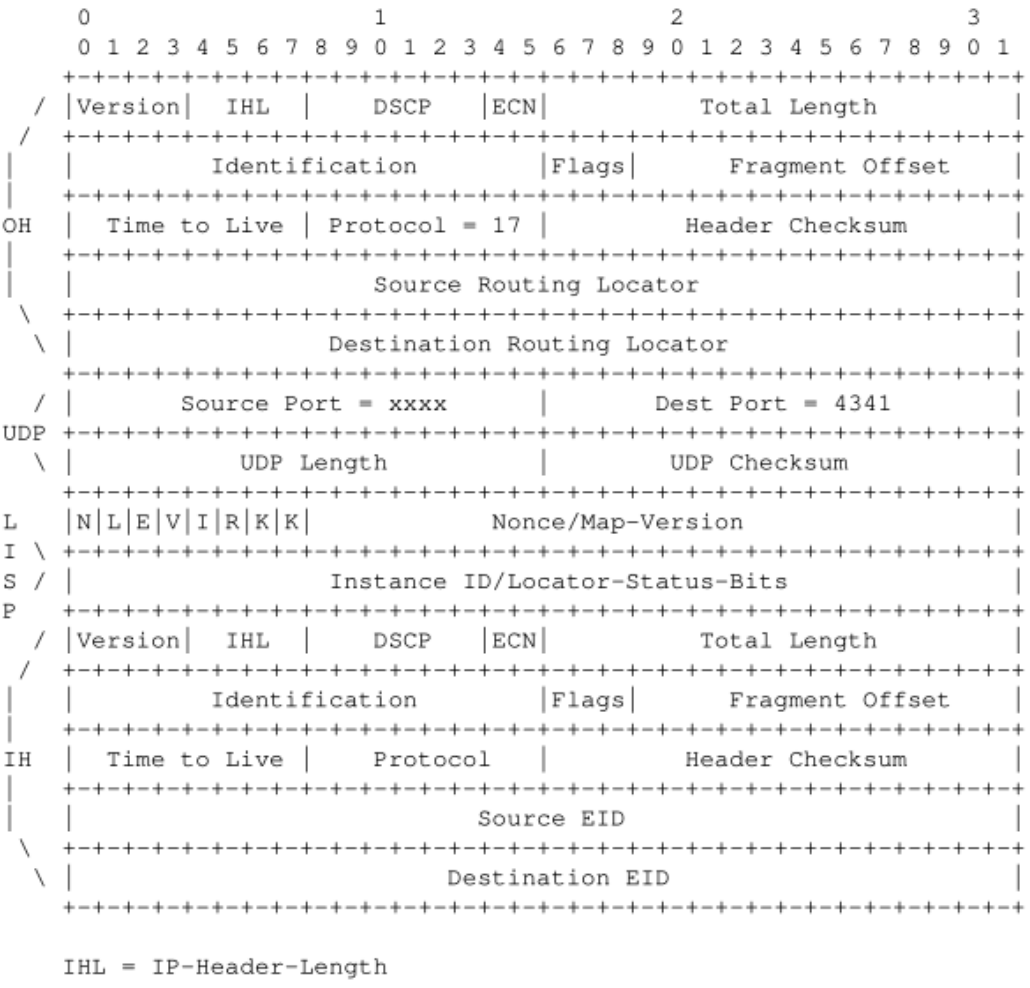

FIGURE 2.2: LISP IPv4-in-IPv4 Header Format.
OH = Outer Header, **IH** = Inner Header.

no meaning in the DFZ, without taking into account the particular case of proxies. To address this situation, the IP packet has to be encapsulated in another IP packet whose addresses are from the RLOC address space. This encapsulation is the bedrock of the LISP data overlay network framed amongst the LISP-capable routers.

This introduced encapsulation is performed by prepending a new IP header to the already existing IP packet. This new header is referred as the outer header (**OH** in Figure 2.2). Consequently, the encapsulated IP header is the inner header. An UDP and a LISP header accompany the outer IP header in order to ensure the good forwarding of this new datagram on all the middleboxes. A middlebox is a networking device whose function is beyond the simple packet forwarding (*e.g.* NAT, Firewall, Proxy, ...) [20]. Some of them might have policies that reject packets that use custom protocols ; it is therefore a good practice to use UDP or TCP in the design of a protocol, if possible [21]. The format of the encapsulated packet's headers is specified by Figure 2.2, for the case of IPv4. The **Protocol** value set to 17 in the outer IP header means that an UDP header should follow ; this is indeed the case. By contrast, the **Protocol** value of the inner IP header is obviously not defined, as it depends on the payload of the sender. The encapsulation is also possible with IPv6 and the format is similar to the IPv4 one. It is worth noting that the same consideration can be made for the IPv6 **Next Header** field, which is similar to the IPv4 **Protocol** field [22].

Thus, the outer IP header contains the source and destination RLOCs of the involved xTRs. The destination UDP port is set to 4341, which refers to a LISP Data Packet. The UDP checksum of the outer header should be set to zero for performance reasons, as the integrity of the payload will anyway be handled by the inner header (**IH** in Figure 2.2). The LISP header contains, among other things, reachability information – providing a way to keep track of the Locator status – and an instance ID – in order to potentially segregate the LISP traffic into different instances.

Let us examine in more details the different fields of the LISP header. As one can notice in Figure 2.2, the header first contains a list of flags followed by the **Nonce/Map-Version** and the **Instance ID/Locator-Status-Bits** fields.

- The **N**-bit (Nonce-Present) is used to determine if a nonce is present in the **Nonce/Map-Version** field. The usage of a nonce will be explained when introducing this field.
- The **L**-bit (Locator-Status-Bits field enabled) determines if the **Locator-Status-Bits** field is in use.
- The **E**-bit (Echo-Nonce-Request) is only used when the **N**-bit is enabled. It requests the xTR receiving this packet to echo back the nonce value.
- The **V**-bit (Map-Version present) is used to determine if database Map-Versioning information is contained in the **Nonce/Map-Version** field. Obviously, the **N**-bit must be disabled if this bit is enabled, and vice-versa. The Map-Versioning mechanism is out of the scope of this work and will not be discussed in more details.
- The **I**-bit (Instance ID bit) specifies that the first 24 bits of the **Instance ID/ Locator-Status-Bits** are reserved in order to contain an Instance ID. In that case, the **Locator-Status-Bits** field is reduced to 8 bits.

- The **R**-bit (Reserved) has no current purpose.
- The **KK**-bits are used when the encapsulated packet is encrypted using LISP data plane confidentiality mechanisms [23].
- The **Nonce** field contains a random value used for the Echo Nonce Algorithm. The goal of this latter algorithm is to determine the reachability between an ITR and an ETR. Upon reception of a LISP packet containing a Nonce (and with both the **N** and **E** bits activated), the ETR will forward the encapsulated packet as usual. Additionally, the ETR will send a new LISP packet containing the same Nonce value (and with the **N** bit enabled). Thanks to this mechanism, the ITR can determine if the targeted ETR is reachable.
- The **Locator-Status-Bits** field is used by the ITR to indicate to the destination ETR the reachability status (up or down) of the different ETRs associated to the source LISP site. Each bit of this field is then associated with the RLOC of an ETR.

In conclusion, the ITR serves the role of prepending the described headers for packets from inside of its LISP site to the outside. The ITR resolves the destination locator by performing an EID-to-RLOCs mapping lookup, using the control plane features. In return, the ETR must strip those headers when receiving LISP Data packets whose destination is one of its RLOCs and forward the stripped packet into its LISP site. This mechanism between the ITR and the ETR constitutes the LISP tunneling over the core of the Internet.

2.1.3 Control plane

As one can notice, the LISP data tunneling strongly relies on the possibility for an ITR to figure the mappings out from the EID space to the RLOC space. Independently of its implementation, the control plane has to offer the possibility for an ETR to advertise and register the EID subspace this device is responsible for. The control plane is also responsible for answering an ITR that has requested information about the mapping of a certain EID.

LISP introduces two devices that constitute the interface between the xTRs and the mapping system : the Map-Resolver and the Map-Server [6]. Thanks to this architecture, the xTRs are able to interact with the mapping system while being oblivious to its implementation. This offers a modularity in the choice of the mapping system and the choice of its design.

The Map-Resolver is a device that listens for mapping requests and answers with the appropriate mapping if it exists, or a negative answer otherwise. In turn, the Map-Server is a device that learns mapping entries from the ETRs by use of a registration mechanism explained below. It handles the propagation of the information within the mapping system.

LISP defines a certain kind of LISP UDP packet in order to communicate with the control plane : the LISP Control Message [6]. Several different types of LISP Control Message coexist, each having an individual specification. Although different, they all begin with a 4-bit long **Type** field needed to distinguish them. The remaining part

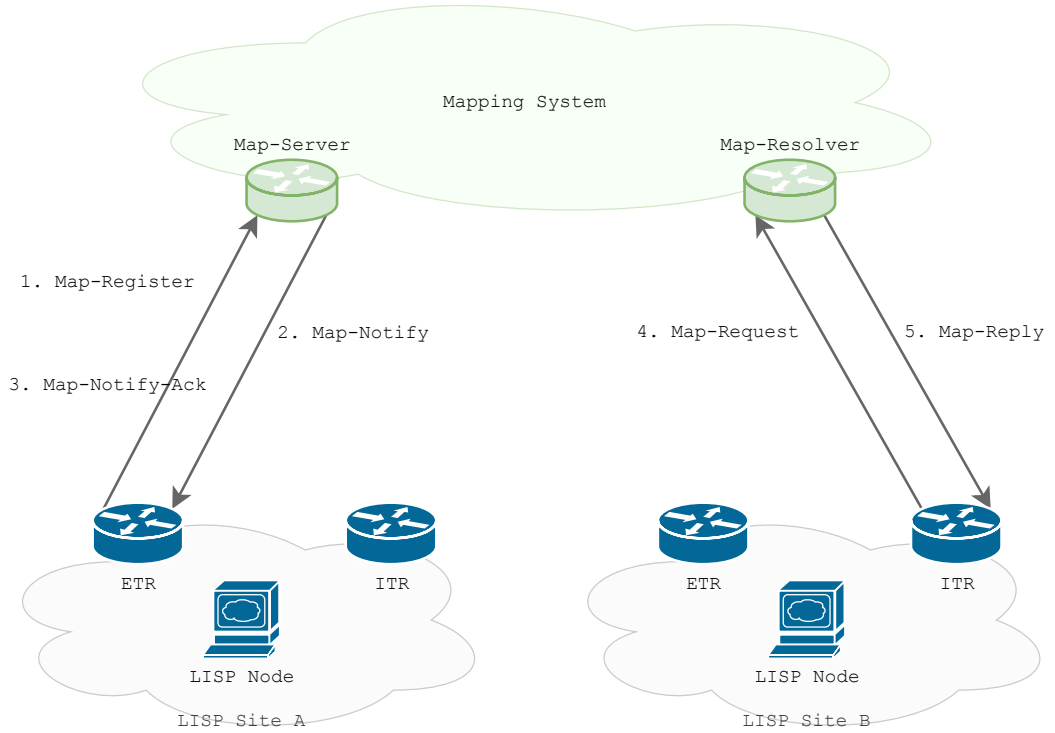

FIGURE 2.3: Overview of the different messages exchanged between the xTRs and the control plane interface.

of the message is therefore specified depending on the message type. A LISP Control Message type can either be a `Map-Request`, `Map-Reply`, `Map-Register`, `Map-Notify`, `Map-Notify-Ack` or an `Encapsulated Control Message`.

Example scenario

Before diving deep into the details of each message type, let us introduce their usage in an example scenario. Figure 2.3 presents a simple setup of LISP nodes and a common use case. The mapping system is represented as a cloud because its implementation is not relevant for using the different LISP Control Messages, as stated before. Moreover, two different LISP sites (`LISP Site A` and `LISP Site B`) are represented and the goal of this scenario is to make the LISP node of each site communicate with each other. It is important to notice that the xTRs are the LISP devices at the boundary of the LISP sites whereas the Map-Register and Map-Server are the ones at the boundary of the mapping system architecture.

First, both LISP sites must advertise the different EID ranges they are responsible for. The exchange of messages between `LISP Site A` and the Map-Server serves this purpose. It should be underlined that a similar exchange has to happen between `LISP Site B` and a Map-Server. As one can notice, this process consists of three different LISP Control Messages : the `Map-Register`, the `Map-Notify` and the `Map--Notify-Ack` messages. The `Map-Register` message is sent by the ETR and contains the information on the EID-prefixes managed by the LISP site. As the name of this message indicates, its purpose is to register mappings into the mapping system. Each registration contains a *time to live* and thus, a LISP site has to send new `Map--Register` messages on a regular basis. Note that a LISP site is allowed to register

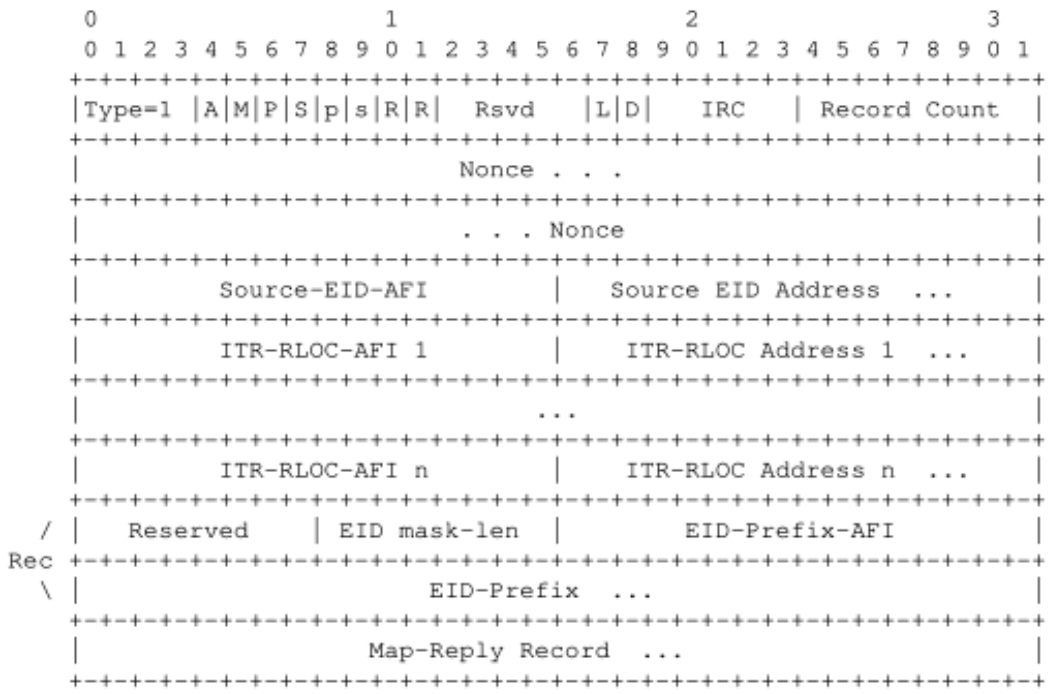

FIGURE 2.4: LISP `Map-Request` message format.
Rec is the portion of the containing information about a record. The number of records in a packet is set by **Record Count**.

mappings to multiple different Map-Servers. Upon reception of a `Map-Register` message, a Map-Server must send back a `Map-Notify` message to either confirm or negate the registration. An ETR can choose not to receive such a message by stating it in its `Map-Register` message. Finally, an ETR can potentially send back a `Map--Notify-Ack` message in order to acknowledge the good reception of a `Map-Notify` message.

Once both LISP sites are well registered into the mapping system, they should be able to retrieve the locator (*i.e.* the RLOC) of each other in order to set up the LISP tunneling. In that respect, the `Map-Request` and `Map-Reply` message types are used to retrieve the required EID-to-RLOCs mappings from the mapping system. An example of this is shown in Figure 2.3 between the ITR of `LISP Site B` and the Map-Resolver. In order to send a packet to a node of `LISP Site A`, the aforementioned ITR has to know the RLOC of one of the ETR of `LISP Site A`. This information is indeed stored in the mapping system. Hence, the ITR will send a `Map-Request` message to a Map-Resolver. This `Map-Request` contains the EID-Prefix on which the ITR needs to know the RLOCs. In this situation, the EID-Prefix is of course one handled by `LISP Site A`. Upon reception of a `Map-Request` message, the Map-Resolver will retrieve the relevant EID-to-RLOCs mapping and send it back to the ITR via a `Map-Reply` message. If no relevant mapping exists, the Map-Resolver will send a Negative `Map-Reply` message.

In-depth description of the message types

Now that the usage of the different LISP Control Message types is overviewed, let us define them more precisely.

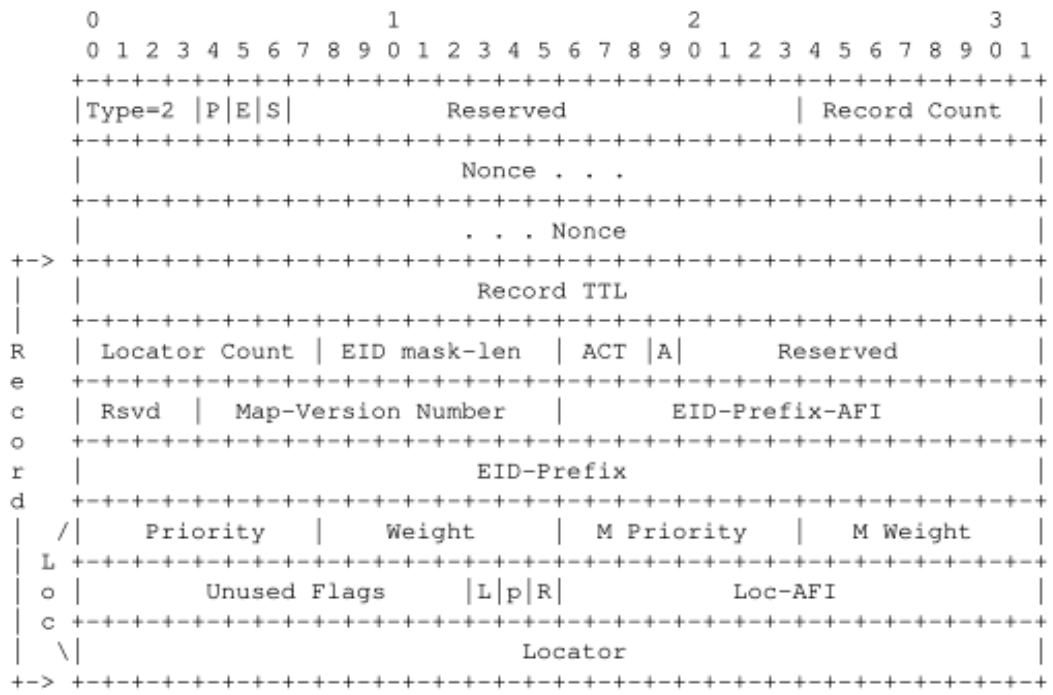

FIGURE 2.5: LISP `Map-Reply` message format.
Loc contains information of a locator entry. **Locator Count** sets the number of locator entries in a **Record**.

The LISP `Map-Request` – the first type of control message – is mainly used to query the mapping system. Most of the time, this packet originates from an ITR requesting a certain mapping to a Map-Resolver. Figure 2.4 presents the UDP payload of the `Map-Request`. Among the fields of this message, the most important is the **EID-Prefix**, which is part of the record, determining the prefix to query. Additionally, the **EID-Prefix-AFI** determines the type of prefix (IPv4 or IPv6) contained in the mentioned field and the **EID mask-len** field determines the number of relevant bits in the **EID-Prefix** field. It is possible to include multiple records in a `Map-Request`, and thus request multiple mappings in one packet. The **Record Count** determines the number of records in the current `Map-Request`.

If the **A** (Authoritative) flag is set to 1, it means that the ITR specifically wants to receive the reply from an authoritative ETR of its requested prefix instead of receiving the reply from the mapping system. In this case, the list of **ITR-RLOC Addresses** can be used to give the possibility to the ETR to select the destination of the reply. Note that this list must contain at least one element, the number of additional entries to this list being defined by the **IRC** field.

One can notice that a **Nonce** field is present in the message. Its value must be randomly generated. This nonce is used to uniquely identify the `Map-Request` and to detect which `Map-Reply` is related to it.

Finally, the **Source EID Address** contains the EID of the host sending the `Map-Request`. This information is optional : the **Source-EID-AFI** can be set to 0 in order to indicate that there is no **Source EID Address** in the message.

The LISP `Map-Reply` is the control message used as a response of a `Map-Request`. It can either originate from the mapping system or from an authoritative ETR. Figure 2.5 presents the format of such a message. This message format provides a structure to send a list of records (whose length is defined by **Record Count**). A record associates an **EID-prefix** to a set of locators ; this is indeed the information contained in an EID-to-RLOCs mapping. One can notice that a **Priority** and a **Weight** is linked to each **Locator**. A low priority value takes precedence over a higher one. The weight can be used to balance the traffic between the different ETRs of the same LISP site.

Let us determine some important fields contained in this message type.

- The **Nonce** field carries the same nonce value as the corresponding `Map-Request`, as explained before.
- The **EID-Prefix-AFI** is used to define the type of EID prefix that is used. In our case, this type is either set to 1 for IPv4 or 2 for IPv6.
- The **L** (Local) flag of a locator entry is set to 1 if the locator in question corresponds to a RLOC reaching the current sending ETR.
- The **R** (Reachable) flag of a locator entry is used to determine whether the relevant RLOC is currently reachable or not. Therefore, it is important to notice that the ETR can advertise unreachable locators.
- In the same fashion than the **EID-Prefix-AFI**, the **Loc-AFI** determines the value type used in the **Locator** field.

If the requested EID prefix is not bound to any locator, a Negative `Map-Reply` is sent. This is simply represented by a `Map-Reply` with an empty set of locators ; and thus, a **Locator Count** is set to 0.

The third type of control message specified by LISP is the `Map-Register` message. It is sent by an ETR to a Map-Server in order to register one or multiple **EID-Prefixes** this ETR is responsible for. The ETR can also include in the `Map-Register` the RLOCs of the other ETRs that are responsible for the same prefixes. Figure 2.6 presents the `Map-Register` message format.

A `Map-Register` message contains fields providing **Authentication**. One or multiple pre-shared secrets have to exist between the ETR and the Map-Server on which the ETR wants to register. A trust relationship is therefore required between both sides. This prevents a rogue to impersonate the ETR with the aim of defining illegitimate mappings. The **Key ID** field is used to select which pre-shared secret will be used to derive a key for the current message authentication, with the appropriate algorithm, defined by **Algorithm ID**. The **Authentication Data** therefore contains a Message Authentication Code (MAC) of the entire message, from the **Type** field until the end[1]. Note that the **Nonce** field is a value that is incremented for each `Map-Register` and is used to avoid replay attacks.

[1] At the time of the generation of the MAC, the **Authentication Data** field has to be preset to 0.

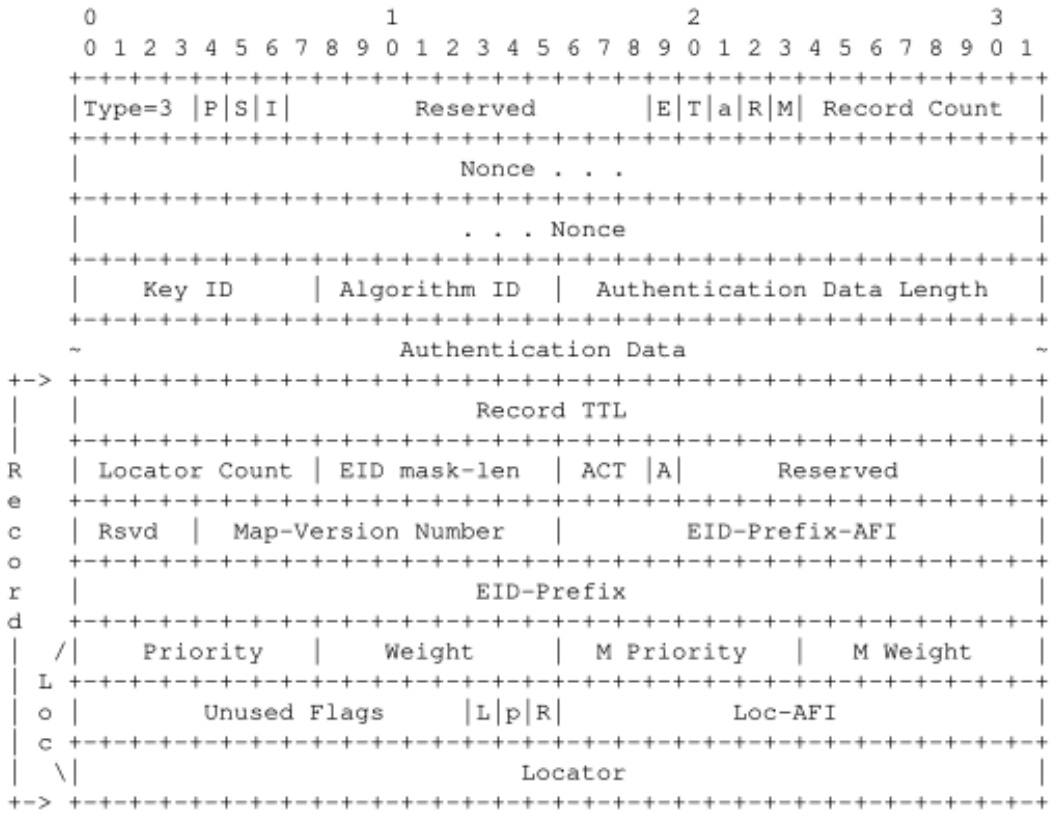

FIGURE 2.6: LISP `Map-Register` message format.
Loc contains information of a locator entry. **Locator Count** sets the number of locator entries in a **Record**.

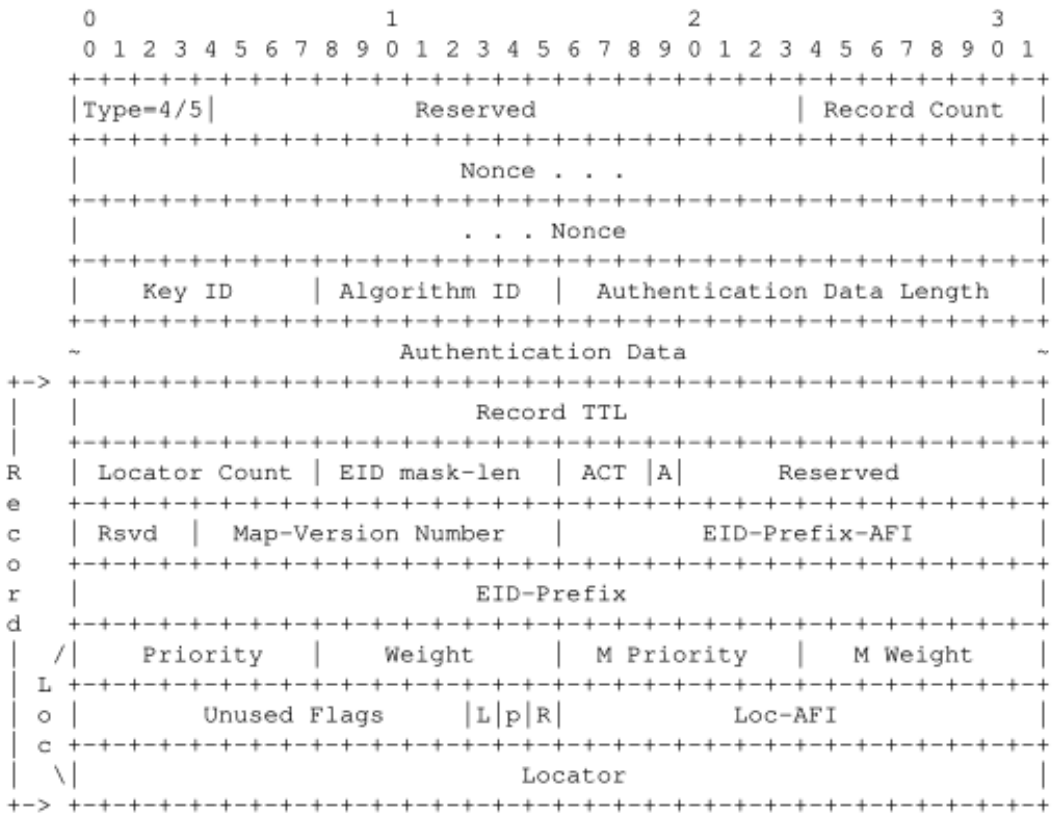

FIGURE 2.7: LISP `Map-Notify` and `Map-Notify-Ack` message format. **Loc** contains information of a locator entry. **Locator Count** sets the number of locator entries in a **Record**.

The **Record** part of the message format is similar to the one of a `Map-Reply` message. Those records contain the mappings that the ETR wants to register as well as the list of Locators authoritative to those prefixes.

Let us mention the **P** (Proxy) flag. If set to 1, the ETR specifically requests that it is up to the Map-Server to assume the responsibility of sending `Map-Reply` messages instead of being up to the ETR to do so. As for the **S** (security-capable) flag, it indicates that LISP-SEC [8] is in use ; this topic will be introduced later on in this work.

Figure 2.7 presents the format of `Map-Notify` messages as well as `Map-Notify-Ack` messages, depending on the **Type** value. The `Map-Notify` message is used to confirm to an ETR the good registration following a `Map-Register` message. This `Map-Notify` is only sent if the ETR had set the **M** (want-map-notify) flag in its `Map-Register` message. The other fields of the `Map-Notify` are copied from the `Map-Register`. This allows the ETR to acknowledge the validity of the information.

A `Map-Notify-Ack` message is used to acknowledge the good receipt of a `Map--Notify` message. This allows the Map-Server to become aware of the reception of the `Map-Notify` by the ETR and therefore to stop trying to resend the message.

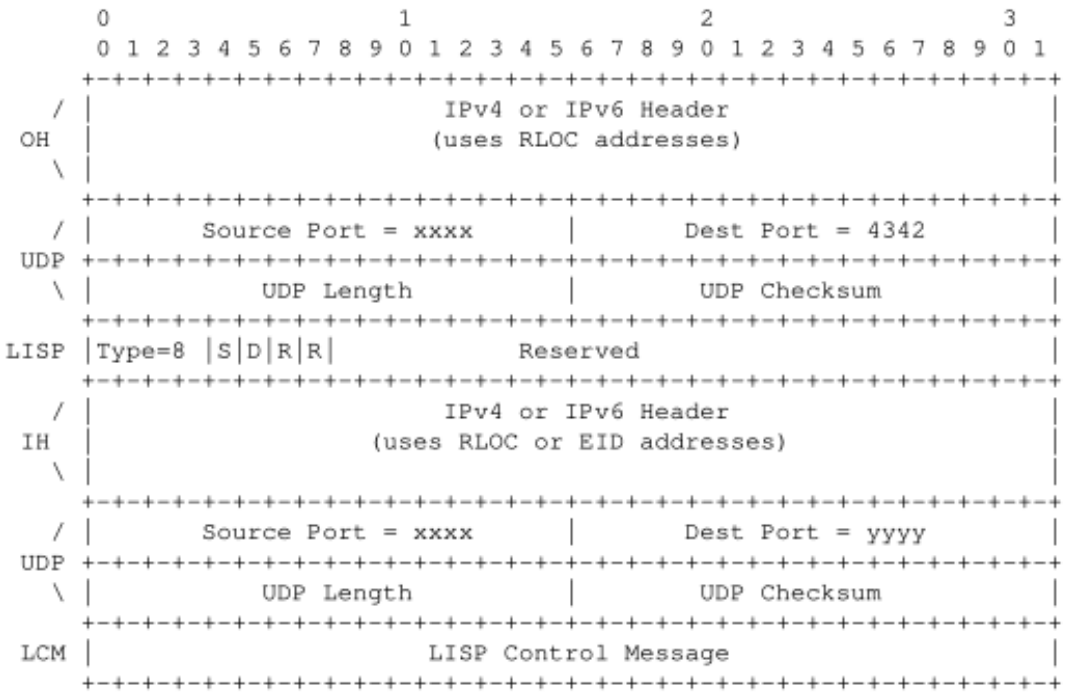

FIGURE 2.8: LISP `Encapsulated Control Message` format.
OH: Outer Header, **IH**: Inner Header, **LCM**: LISP Control Message.

The last type of control message specified by LISP is the `Encapsulated Control Message` (ECM). This message is characterized by the **Type** field set to 8. This message contains fewer fields than the other control messages. The only fields present are flags, as one can notice on Figure 2.8. This message is used to encapsulate another control message in it. Most of the time, the ECM is used to encapsulate a `Map-Request`. In this case, such a message is called, by extension, an `Encapsulated Map-Request`.

The `Encapsulated Map-Request` is often used by an ITR to request mappings to a Map-Resolver, instead of simply sending a plain `Map-Request`. Indeed, depending on the need of the mapping system implementation, the `Map-Request` can possibly contain different source and destination address values than the expected ITR and Map-Resolver RLOCs[2]. In such a case, this `Map-Request` has to be encapsulated in an `Encapsulated Control Message` to allow the good forwarding of the message, by setting the aforementioned RLOCs in the outer IP header.

The use of an `Encapsulated Control Message` instead of a simple `Map-Request` also permits the mapping system to forward the `Map-Request` (originated from an ITR) to the corresponding ETR while leaving the whole `Map-Request` content intact, including the IP and UDP layers. This enables the system to implement authentication and integrity mechanisms on the `Map-Request` messages. LISP-SEC [8] implements those security aspects and can be enabled by setting the **S** (Security) bit of the `Encapsulated Control Message`. In this case, the **Reserved** field is used to contain authentication data. The security topic is discussed in more details further on in Chapter 3, page 28.

[2]Even more, the LISP specification also allows a `Map-Request` to contain addresses from the EID space in its IP header.

Note that both **UDP Checksum** fields of an `Encapsulated Control Message` must be non-zero, if the checksum is incorrect, the control message must be dropped. In the case of an `Encapsulated Map-Request`, the inner destination port also has to be set to 4342, which is the LISP Control Packets port. The **D** (DDT) flag is reserved for the use of LISP-DDT [7].

Let us point out that other **Type** values are reserved but not specified by LISP. This is the case of the **Type** value 6, defining a LISP `Map-Referral` message. This type of message is used and specified by LISP-DDT [7], the currently deployed implementation of the mapping system. LISP-DDT is explained in details in Section 2.2.3.

2.1.4 Conclusion

As we have seen above, the Locator/Identifier Separation Protocol (LISP) addresses the growing scalability concerns about routing on Internet by separating the locator space from the identifier space. In this way, the DFZ is oblivious to the identifier of each node but will instead only have information regarding the location of the site in which the node is situated. The locator space can therefore be managed so as to optimize the routing.

In order to make it possible for two LISP sites to communicate between each other, a tunneling has be established between both sites. Indeed, as the EID addresses are irrelevant in the core of the Internet, packets whose addresses belong to the EID space must be encapsulated in packets containing RLOC addresses. This is the purpose of the LISP data plane.

Whenever performing a LISP encapsulation, one has to be aware of the EID-to-RLOCs mappings, in order to be able to translate an EID address into a RLOC one. The role of the control plane is to gather this mapping information and make it accessible. The control plane is designed as a pull model : the entity needing information about a mapping has to request it from the control plane. In contrast, a push model would have required the control plane to spread the mapping updates to all the entities.

LISP is specified in such a way that the implementation of the mapping system – the database containing the mapping information – is flexible and modular. Only the interface between the data plane and the control plane is specified by LISP. In practice, many mapping system architectures were proposed, all having different design choices. In Section 2.2, the most notable mapping systems are presented.

2.2 The LISP mapping system

2.2.1 Introduction

The mechanism of the LISP protocol has now been introduced. However, the denial-of-service attack that will be designed in Chapter 4 more precisely relies on the LISP control plane and its mapping system. In that sense, it is important to analyse the different possible implementations of the mapping system and focus on the one that will be used throughout the attack.

This section is devised into two parts. First, a broad review of the choices that one can make in order to design a mapping system will be done. In that regard, a taxonomy of mapping system implementations will be defined. The second part details the implementation of LISP-DDT [7], the *de facto* standard mapping system as for today.

2.2.2 Taxonomy

Before entering into the details of the mapping system implementation that is used throughout this work (LISP-DDT), let us have a broad overview of the different possible implementations that exist. In order to do so, instead of simply going over the different notable mapping systems that are being developed, let us define a structure allowing us to classify them according to their design. This will allow us to clarify the differences between each of them in an overall perspective.

A general taxonomy for mapping systems was proposed by Hoefling and Hartmann [2]. First and foremost, it is important to note that the use of this particular taxonomy was chosen in an arbitrary manner by the author of this work in order to provide a structure for the different mapping systems that are mentioned. Other taxonomies for mapping systems could have been used, such as the Ramirez *et al.* taxonomy that classes them in four different categories (DNS based, DHT based, Distributed and Routing based) [24]. Anyhow, let us describe the selected taxonomy in order to use it for classifying the several different mapping systems outlined in this section.

As a reminder, the mapping system is a database of EID-to-RLOCs mappings. More precisely, and in order to avoid having a mapping for every single EID, the mapping system will map an EID prefix to one or many RLOCs. Thus, the EID addresses can be aggregated into prefixes. Note that it is done in a way that prefixes with the longest length – *i.e.* the most specific ones – take precedence [5]. This is comparable with the way a forwarding table is built, for the network layer [25].

A node of the mapping system in use – referred as a map-base (MB) by the authors of the taxonomy – can either have a full knowledge (MBFK) about all the mappings of the system or, inversely, only have a partial one (MBPK). This difference constitutes the first junction of this mapping systems taxonomy. An overview of the whole taxonomy is available in Figure 2.9. Examples of notable mapping system implementations are presented along the different taxa defined in this section.

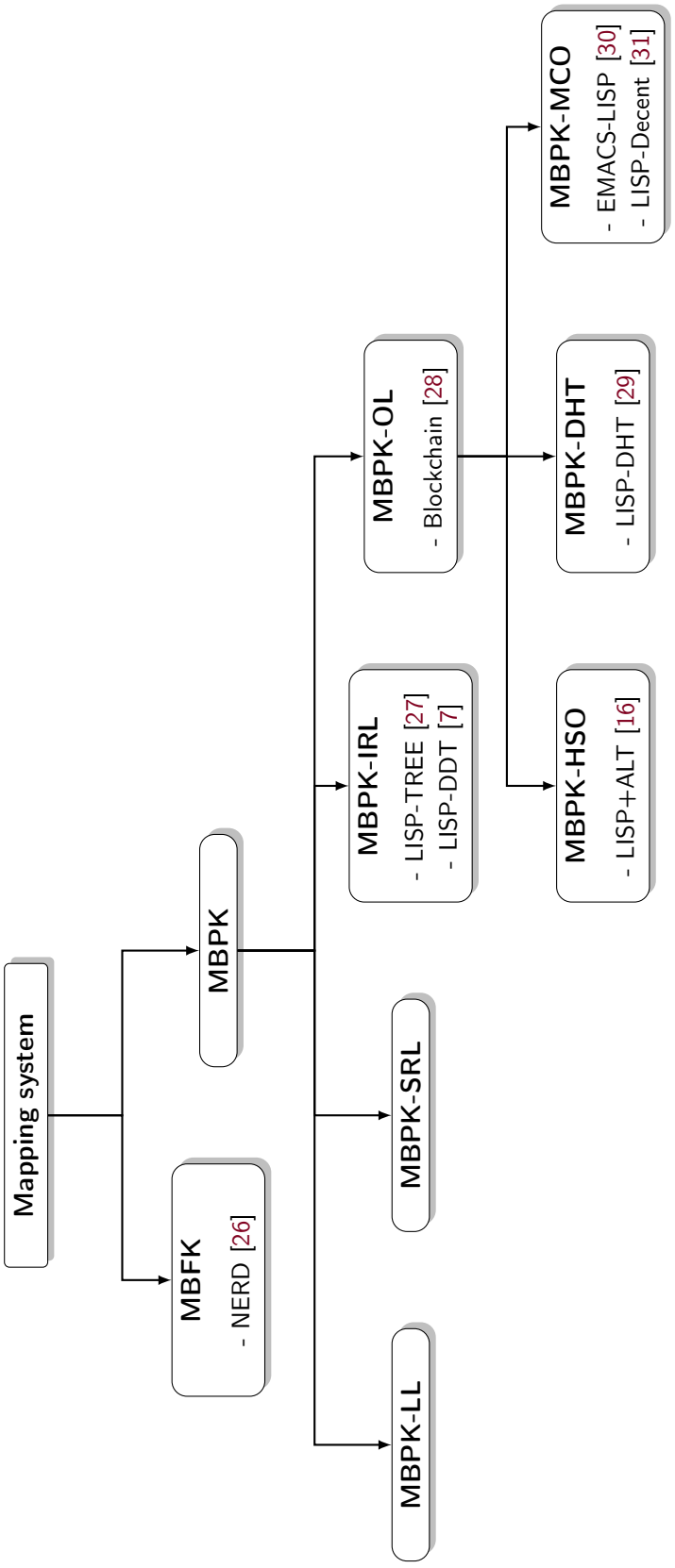

FIGURE 2.9: Hierarchical taxonomy of mapping systems [2].
Notable mapping system implementations are included in the taxa.

With the **full knownledge** case, all map-bases have to contain the same information and maintain an overall consistent state. Even with prefix aggregation, we can imagine that the size of such a database will grow according to the scale of the system. Moreover, a mechanism has to be put into place in order to advertise every map-base of a possible update in the mappings. Depending on the number of map-bases in the system, this can potentially lead to a significant traffic. Therefore, we can sense that this design is not suited for systems needing frequent changes in the mappings. Nevertheless, the query of a mapping is really quick and simple as every map-base contains this information on their own. Thanks to this aspect, one can be sure that no packet has to be dropped while requesting the relevant mapping for the encapsulation. The NERD (Not-so-novel EID-to-RLOC Database) is an example of such a mapping system implementation [26]. In the latter, the mappings are centralised in database authorities, which are then replicated in every ITR.

The implementation of map-bases with **partial knowledge** is not as straightforward knowing that the system has to be able to locate where a particular mapping information is situated, one way or another. Multiple solutions coexist for making the lookup possible. As shown in Figure 2.9, this taxonomy describes four different discovery options in order to do so : MBPK-LL, MBPK-SRL, MBPK-IRL and MBPK-OL.

The first option presented is the **local lookup** (MBPK-LL on Figure 2.9). In this configuration, it is up to the ITR to maintain a table storing EID-prefix-to-MB information ; with as much prefix aggregation as possible to keep this table small. In this regard, the lookup is really fast because the ITR directly knows which map-base to contact in order to receive a certain mapping. Assuming that the mapping system architecture is quite stable, the information stored in the ITR does not change that often. Nevertheless, a solution has to be implemented in order to keep this information up to date.

Instead of keeping the EID-prefix-to-MB mappings locally (as it is the case with MBPK-LL), this information can be stored in a remote device, serving as global authority. This situation represents the second discovery option, the **single remote lookup** (MBPK-SRL on Figure 2.9). Compared to the MBPK-LL, this option has to perform one more lookup in order to finally obtain the desired mapping from the system. The first lookup contacts the featured global authority to know on which device (a map-base) the ITR has to perform the second lookup, and obtain the desired RLOCs. Note that the MBPK-IRL can possibly store the information in a local cache after the lookup to increase performances.

The **iterative remote lookup** option (MBPK-IRL on Figure 2.9) extends the principle of MBPK-SRL in authorities divided into multiple levels. Hence, for a certain EID-prefix, the ITR first contacts the level-0 authority that itself answers back a pointer to the appropriate level-1 authority. The ITR digs the layers iteratively until receiving a pointer to the relevant map-base, the leaf of the tree. The Domain Name System (DNS) [18] is a textbook case of such an architecture. Just as for the MBPK-SRL option, the MBPK-IRL can cache the information obtained from the different authority devices. In the case of LISP mappings, two notable implementations stand out. The first one, LISP-TREE, actually relies on the DNS implementation for its infrastructure [27]. The records stored in LISP-TREE consist of locators for the

different ETRs of the system. This mapping system benefits from the maturity of the DNS and is also able to use DNSSEC [32] in order to provide security in the system. The second mapping system implementation listed in this taxon is LISP-DDT [7]. Although LISP-DDT also drawn inspiration from the DNS, this mapping system specifies a completely custom protocol. LISP-DDT is the mapping system that is currently deployed in the LISP Beta Network [33]. Hence, the attack that will be presented in Chapter 4 implies the fact that LISP-DDT is the mapping system in use. For that reason, this implementation is described in details in Section 2.2.3.

Finally, the last option of a partial knowledge solution – the **overlay lookup** (MBPK-OL on Figure 2.9) – is to design the system by mean of an overlay network. The goal of this overlay network is to forward the `Map-Request` from the ITR to the appropriate map-base (*i.e.* an ETR or a proxy device). Let us mention that the reply does not have to travel the overlay network ; the answering device is able to directly respond to the querying ITR. Of course, the ITR has to be attached with at least one entry node of the overlay network, however the latter is designed. The overlay network can be practically constructed in various fashions, one can cite :

- The **hierarchically structured overlay** (MBPK-HSO), whose nodes form a tree. Each node manages a certain EID-prefix and delegates the duty to its children – managing subsets of this prefix – or a map-base. The map-bases connect with the overlay node that handles the most specific EID-prefix relevant to the map-base in question. This lookup option is similar in design with the MBPK-IRL, with the difference that a `Map-Request` can travel up the tree if the entry point is in another branch. Another main difference is the fact that, here, the lookup can not only be done iteratively but also recursively or in a hybrid fashion.

- The **distributed hash table** (MBPK-DHT) option, on which the different map-bases form a distributed hash table. Each node is attributed an order number. A hash-function on the queried EID-prefix is used to determine which map-base is responsible to answer it. Furthermore, a specific map-base should also be able to forward a request to, at least, the map-base following itself, in order. In this way, a request can be injected in the system from any node and can still eventually reach the relevant map-base.

- The **multicast overlay** (MBPK-MCO), which takes advantage of the multicast communication. A multicast group is created for each determined subset of the EID space. The different ETRs are subscribed to the multicast groups relevant to them. In this way, an ITR needing a certain mapping will send the request to the multicast group associated to the relevant EID space subset ; the intended ETR will sure enough receive the `Map-Request` sent by the ITR.

An example of hierarchically structured overlay is the LISP+ALT [16] mapping system, the first one to have been deployed. This system uses an overlay of routers tunneled between each other and running BGP (Border Gateway Protocol) [17] in order to advertise the EID-prefixes they are responsible for. Thanks to this, a `Map-Request` message entering the overlay is routed to the relevant device.

In the case of the distributed hash table option, LISP-DHT [29] is a good example of mapping system implementation relying on this. All the nodes responsible for mappings connect into a Chord ring and are identified according to the prefix they

manage. Chord is a popular protocol for peer-to-peer distributed hash tables [34]. A request for a certain mapping is forwarded inside the ring until reaching the appropriate node.

Regarding the multicast overlay option, LISP-Decent [31] and EMACS-LISP (EID Mappings Multicast Across Cooperating Systems for LISP) [30] are two notable candidates. The first one, LISP-Decent (in a push-based mode), suggests to create multiple multicast groups and broadcast the `Map-Request` messages in them in order to reach the device that is responsible for the requested prefix. Note that a pull-based mode also exists in LISP-Decent. In that case, the lookup is more comparable to an iterative remote lookup one ; indeed, the list of Map-Servers managing the prefixes is stored as records in the actual DNS. On the other hand, EMACS-LISP, the second mapping system implementation of this taxon, takes advantage of multicast groups in order to directly broadcast the data packets (instead of the control messages) whenever the mapping is not already known by the sending ITR.

Finally, it is important to notice that it is possible to find mapping system implementations that cannot be classified in the currently used taxonomy. For instance, a mapping system based on the blockchain technology can possibly be implemented [28]. In such a system, the mapping information is stored in the ledger of the blockchain. This system has the advantage of not needing to manage certificates in order to enable data authentication [35].

As we just have seen, many different possibilities coexist in order to design the EID-prefix-to-RLOCs mapping system. The taxonomy that has just been presented here allows us to have a good notion of the variety of the implementation the mapping system can adopt. Nevertheless, the results presented in this work assume that LISP-DDT is the mapping system implementation in use.

2.2.3 LISP-DDT

The LISP Delegated Database Tree (LISP-DDT) [7] is the second mapping system implementation to have ever been deployed, along with the aforementioned LISP+ALT system. Hence, in this work, we consider that LISP-DDT is the mapping system implementation in use. Moreover, this is also the mapping system that is used in the worldwide LISP network prototype accessible over the public Internet, the LISP Beta Network [33]. The remaining of this chapter will thus be dedicated to describe LISP-DDT in details, as the denial-of-service attack that will be presented in Chapter 4 relies on this system.

LISP-DDT has an architecture that is really similar to the DNS. As a matter of fact, this mapping system works as an iterative remote lookup, to employ the terminology of the Hoefling *et al.* taxonomy. In that respect, this system is composed of DDT nodes hierarchised in a tree. Each DDT node is authoritative for certain XEID-prefixes ; an extension of the concept of EID-prefixes that will be introduced straight after. A DDT node must maintain a list of pointers for all the sub-prefixes of the intervals it is responsible for. In that matter, the children of a node represent the different nodes that are pointed to by the node in question. Inexorably, a root

DDT node – the top of the tree – is covering the whole XEID space ; it is thus authoritative for all the XEID-prefixes. Conversely, the leaf nodes of the tree are the DDT Map-Servers.

An XEID-prefix (Extended EID-prefix) is a notion specified by LISP-DDT itself. As its name indicates, it extends the concept of EID-prefixes by including additional information. This way, it contains :

- a **DBID** (LISP-DDT Database-ID), a number that could potentially be used in the future in order to be able to use multiple different databases in the DDT. At the present time, this value must be set to 0.
- an **IID** (Instance ID), allowing one to use different contexts of EID-prefix. This can for example be useful in the situation of a Virtual Private Network (VPN), where address allocation is specified differently [36].
- the **AFI** of the EID-prefix, as specified by IANA [13].
- the **EID-prefix**, strictly speaking.

It is important to mention that LISP-DDT does not necessarily store mapping information in itself (except for the case of `Map-Reply` proxying). Instead, it provides a way to transport the `Map-Request` message up to the appropriate device, *i.e.* an ETR that has registered the prefix included in the aforesaid message. A DDT client is any device that is able to communicate with the DDT architecture, in an iterative fashion. Such a device handles two kind of messages in order to communicate with the DDT : the `DDT Map-Request` and the `Map-Referral`. Both message types will be described in details in this section. Naturally, a Map-Resolver is a DDT client in the case where LISP-DDT is the mapping system implementation in use. Nevertheless, an ITR can also potentially directly communicate with the DDT without passing through a Map-Resolver. In that specific case, the DDT client is the ITR. For the sake of simplicity, let us consider the fact that a DDT client is a Map-Resolver, if not otherwise stated.

Upon reception of an `Encapsulated Map-Request` message, the goal of the DDT client is to be able to forward it to an appropriate DDT Map-Server. The Map-Server will itself treat the `Encapsulated Map-Request` and act accordingly, either by sending the message to an ETR or by directly answering to the ITR on behalf of the ETR in question. This is indeed the expected operation of a Map-Server, as explained in section 2.1.3. Note that, like for most of the mapping systems relying on an overlay network, the `Map-Reply` message does not cross the LISP-DDT mapping system. Instead, the ETR sends this message straightaway to the ITR. To come back to the DDT client, this device will therefore need to find this Map-Server in the first place. This will be done by performing an iterative lookup on the DDT architecture. Thus, the DDT client will first send a `DDT Map-Request` message to a root DDT node. The `DDT Map-Request` message basically contains the information of the `Encapsulated Map-Request`; this will be detailed afterwards. This root node will answer with a `Map-Referral` message. This message contains a pointer to its child that is responsible for this request. The DDT client will carry on the lookup, and receive as many `Map-Referral` message as needed until receiving a `Map-Referral` message pointing to the intended DDT Map-Server.

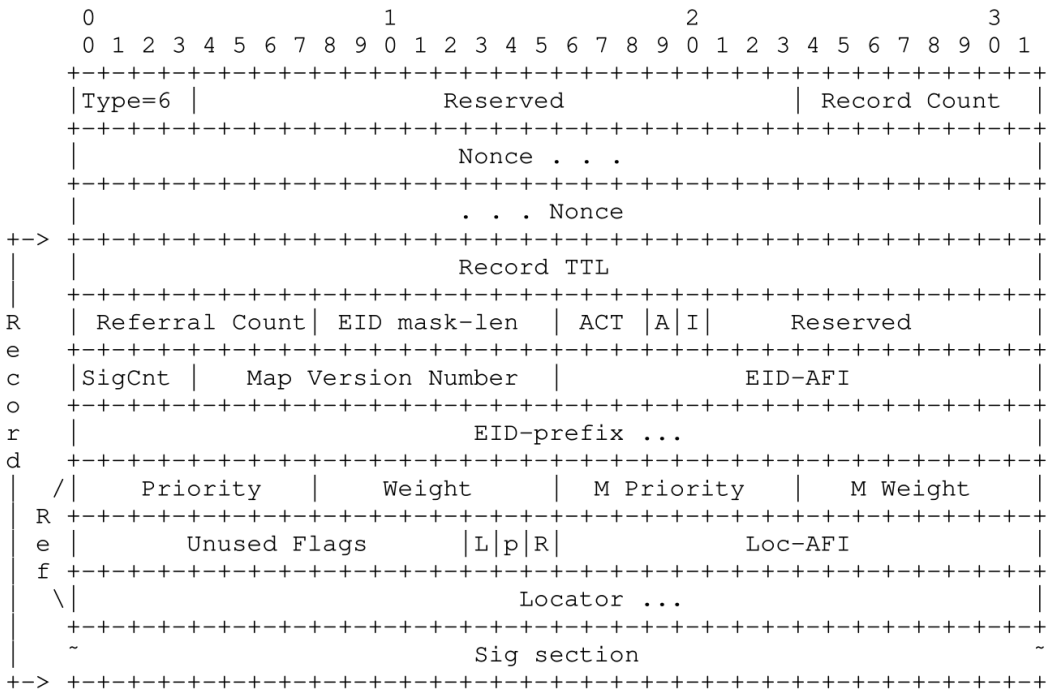

FIGURE 2.10: LISP-DDT `Map-Referral` message format (Type 6).
Ref contains information of an entry in the Referral Set of the **Record**.

Let us point out that the DDT client – the Map-Resolver – can maintain a referral cache of the different DDT nodes and Map-Servers in order to speed up the process whenever a request for an already encountered prefix is treated. The different entries in the cache have a limited time to live. Indeed, the state of the DDT architecture can change over the time.

As we have just seen, the DDT client dug into the DDT tree until finding the Map-Server it needed. This DDT client will now be able to forward the `DDT Map-Request` message to the Map-Server. The latter will in turn be able to recover the `Encapsulated Map-Request` message and proceed. Additionally, the DDT Map-Server will notify the good processing of the request by sending an acknowledgement to the DDT client. This acknowledgement is in the form of a particular kind of `Map-Referral` message. However, in the case where the requested XEID is not registered by any ETR, the Map-Server will inform the DDT client with another kind of `Map-Referral` stating that no ETR is registered for this request. Let us now describe in details the different messages that are used inside the mapping system : the `DDT Map-Request` and the `Map-Referral`.

The format of the `DDT Map-Request` follows the one of a LISP `Encapsulated Control Message`. Therefore, its format can be seen on Figure 2.8, page 15. The difference between both types of message resides in the **D** (DDT-originated) flag ; in the case of a `DDT Map-Request`, this bit is set to 1. In particular, this flag is used by the DDT client in order to indicate to the destination device (a DDT node or a DDT Map-Server) that a `Map-Referral` message must be returned. Besides, the payload of the `Encapsulated Control Message` – the **LISP Control Message** field in Figure 2.8 – obviously contains a `Map-Request` message.

As loosely mentioned in section 2.1.3, the **Type** value 6 of the LISP Control Message format is reserved for the usage of LISP-DDT. As a matter of fact, this value is used to define the second type of LISP Control Message used by LISP-DDT, the `Map--Referral`. Figure 2.10 presents the format of such a message. It basically contains an action code – used to characterise the `Map-Referral` type – as well as the potential information needed for the delegation (*i.e.* the pointer to the next device to contact for the iterative lookup).

The format of the `Map-Referral` message strongly relies on the `Map-Reply` one. Hence, most of the fields contained in the `Map-Referral` is specified by the LISP control plane architecture [6] instead of LISP-DDT itself. For those fields, we invite the reader to refer to the description of the LISP `Map-Reply` message format from section 2.1.3, page 11. Let us new describe the most important fields introduced by LISP-DDT in this datagram.

The record's **ACT** (Action) field is used to define the type of the current `Map--Referral`. As its name indicate, it determines the action the receiver – a DDT client – must take once this message is received. Six possible values for the action code are specified by LISP-DDT.

- The **NODE-REFERRAL** (value 0) is used to indicate that the current device knows at least one DDT node (a child) that is more qualified to find the relevant Map-Server. This indeed means that this child DDT node has a more restrictive XEID-prefix over the request. In the case of a NODE-REFERRAL `Map-Referral`, the message must at least contain one referral in its referral set.
- The **MS-REFERRAL** (value 1) indicates, in turn, that the current LISP node is aware of at least one DDT Map-Server authoritative for the requested XEID-prefix. This action code is really similar in nature to the previous one. Nevertheless, this is informative to the DDT client to know that the next device to contact is actually the Map-Server. Indeed, the security information of the original `Map-Request` should only be transmitted to the Map-Server and not the intermediate DDT nodes.
- The **MS-ACK** (value 2) is the action code used by the Map-Server in order to notify the DDT client that the request has properly been treated. This means that the Map-Server has either forwarded the `Map-Request` message to one ETR that is responsible for the EID-prefix or that the Map-Server has directly answered the ITR with a `Map-Reply` message, as a proxy.
- The **MS-NOT-REGISTERED** (value 3) is used to indicate to the DDT client that the answering DDT Map-Server is indeed responsible for this XEID-prefix but that no ETR is registered for this prefix.
- The **DELEGATION-HOLE** (value 4) indicates that the requested XEID-prefix is cannot be delegated to any DDT node. Hence, this prefix is a hole in the LISP XEID space.
- The **NOT-AUTHORITATIVE** (value 5) can be used by a DDT node in order to indicate that this node is not authoritative for the requested XEID-prefix. This scenario can occur either because of a configuration error in the DDT or because the DDT node used a cached referral that has become invalid meanwhile. In the latter case, the DDT node may possibly decide to erase this entry

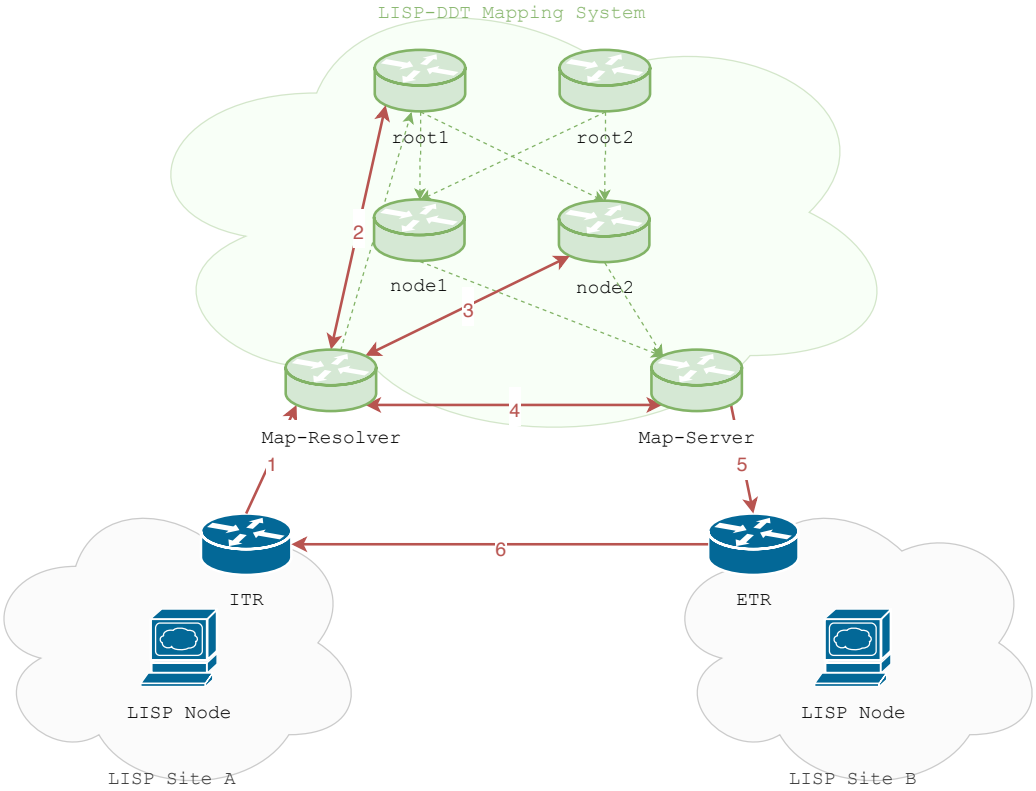

FIGURE 2.11: Overview of a typical mapping request over LISP-DDT. `LISP Site A` and `LISP Site B` should contain at least one ITR and one ETR ; only the devices that are involved in the scenario are represented.

from cache and restart the lookup starting from a DDT root. In the case of a configuration error, the DDT node should silently discard the process of the request.

Most of the case, the DDT node needs to attach referral information along its `Map-Referral` message for the purpose of indicating the locators of the next device to contact in the iterative lookup. This information is included in the Referral Set of the record. The size of this set is determined by the **Referral Count** field. Each element of this set is determined by the different fields enclosed by **Ref** in Figure 2.10. The most notable field is the **Locator** one, which is used to contain the RLOC of the present referral.

Finally, the `Map-Referral` message contains security-related fields. The security aspect of LISP-DDT will be discussed in details in chapter 3. In a nutshell, the security implementation of LISP-DDT is really comparable to DNSSEC [32]. The record of a `Map-Referral` message can be signed in order to provide authenticity and integrity. The relevant signatures are included in the **Sig section** field of the message. The **SigCnt** field indicates the number of signatures that are present.

Now that we have a good insight about the LISP-DDT specification, let us describe a typical mapping request from an ITR using a LISP-DDT mapping system. Figure 2.11 presents a configuration of two LISP sites (`LISP Site A` and `LISP Site`

B) communicating with a LISP-DDT Mapping System. In the presented scenario, the LISP node of `LISP Site A` tries to send an IP packet to the `LISP Site B` LISP node. The shown `LISP Site A` ITR will thus need to encapsulate the packet in order to form a LISP tunnel up to the `LISP Site B` ETR, as explained in section 2.1.2. In order to do so, the ITR will need to become aware of at least one RLOC corresponding to a `LISP Site B` ETR. Hence, the ITR will send an `Encapsulated Map-Request` message to a Map-Resolver (**Step 1**), as detailed in section 2.1.3. Note that we consider, in this situation, that the ETR is well registered to the Map-Server presented in Figure 2.11, for a certain EID-prefix. Moreover, the `LISP Site B` LISP Node has been attributed an ETR address included in that prefix. This is the address that the `LISP Site A` LISP Node uses in order to contact the device.

Let us now consider what is happening inside the Mapping System, which was considered as a blackbox in previous sections. Upon reception of the `Encapsulated Map-Request` message, the DDT Map-Resolver will be charged to find a Map-Server on which the `LISP Site B` is registered. Let us make the assumption that this information is not already listed in the Map-Resolver cache, the latter will thus need to begin the lookup from the root DDT node. In this way, the Map-Resolver sends a `DDT Map-Request` message to a root DDT node (`root1` in the example) which will itself answer with a NODE-REFERRAL `Map-Referral` message containing a list of its children authoritative for this XEID-prefix (**Step 2**). Let us point out that the green dashed arrows in Figure 2.11 represent the static configuration of each device. Thus, we can see that the DDT Map-Resolver is configured with `root1` as root DDT node, for instance.

The DDT Map-Resolver must continue the iterative lookup until reaching a relevant Map-Server. In order to do so, the Map-Resolver will choose between the different nodes listed in the received `Map-Referral` message (`node1` and `node2`). In this example, the Map-Resolver chooses to contact the `node2` DDT node and will send to it, in turn, the `DDT Map-Request` message (**Step 3**). The mentioned DDT node will answer with a MS-REFERRAL `Map-Referral` message, containing at least one locator for the Map-Server of this example (**Step 3**).

At this point, the Map-Resolver is now able to contact the Map-Server, as required. Thereupon, the `DDT Map-Request` message will be sent to the DDT Map-Server, including the potential security information of the initial `Encapsulated Map-Request` message (**Step 4**). The security aspect will be discussed in more details in Chapter 3. The Map-Server will then confirm the correct processing of the request by answering with a MS-ACK `Map-Referral` (**Step 4**). Meanwhile, the Map-Server will send the `Encapsulated Map-Request` message to a relevant ETR (**Step 5**) to continue the process. This behaviour is indeed expected in order to be conform with the LISP control plane specification, presented in section 2.1.3. At this point, the remaining of the operation is independent from the mapping system implementation.

Ultimately, the ETR will receive the `Encapsulated Map-Request` message and will be able to answer to the ITR using a `Map-Reply` message (**Step 6**). It is important to notice that the `Map-Reply` message does not cross the mapping system ; this packet is directly sent to the intended destination. The ITR is now able to create a LISP tunnel towards the `LISP Site B` in order to establish a communication between both LISP nodes. Let us point out that a similar scenario has to happen in a symmetrical

fashion so as to make the `LISP Site B` able to send packets towards the `LISP Site A`. Therefore, the `LISP Site A` needs to have at least one ETR that is registered at a Map-Server of the mapping system. Moreover, the `LISP Site B` requires to have an ITR to query the said mapping system.

In conclusion, the LISP-DDT mapping system provides an easily manageable protocol inspired by the DNS [37]. As we have seen, it consists of different nodes hierarchised into a tree that serves the purpose of directing the `Encapsulated Map-Request` messages up to the relevant Map-Servers. As for today, LISP-DDT is the mapping system implementation that is currently used by the LISP Beta Network [33]. Moreover, this is the mapping system that is used during the denial-of-service attack presented in Chapter 4.

2.2.4 Conclusion

As we have seen in this section, due to the modularity of LISP, its mapping system can be implemented in a variety of ways. Currently, the most mature implementation is LISP-DDT, in which the lookup is done in an iterative remote fashion. Although the architecture of this mapping system is comparable to the one of DNS, LISP-DDT specifies a new protocol, extending the one of the LISP control plane. This section covered the specification of this protocol in order to later find out possible vulnerabilities on which to base an attack. Before doing so, it is important to analyse the security characterized in the LISP-DDT protocol as well as the LISP security in general.

Chapter 3

LISP mapping system trust model and security aspects

3.1 Providing security in LISP

3.1.1 Introduction

The previous chapter presented the different specifications and features of LISP and the LISP-DDT mapping system. As LISP can potentially be deployed in untrusted environments such as the public Internet, security features must be implemented in all aspects of the protocol.

Depending on the messages exchanged and the different nodes involved, one must need to provide security features such as confidentiality, integrity, non-repudiation, authentication, ...

Security can be implemented in the data plane as well as in the control plane. Because of the fact that the denial-of-service attack presented in Chapter 4 operates on the control plane, this chapter mainly describes the latter's security.

Yet, in regard of the data plane, confidentiality can easily be provided along the LISP encapsulation as well as authentication and message integrity [23]. The different xTRs (ITR or ETR) compute a shared secret between each other for the encryption of their future messages by using the Diffie-Hellman key exchange [38]. The different Diffie-Hellman parameters thus have to be exchanged between both xTRs. This is done by using the control plane, using `Map-Request` and `Map-Reply` messages on which the records are encoded with a LCAF type provided for that purpose : the LISP Security Key LCAF Type [14].

In the case of the control plane, multiple security aspects have to be included. Mainly, the registration of the EID-prefixes has to be authenticated between an ETR and a Map-Server in order to ensure that the ETR is indeed allowed to register the prefix at issue. In addition, an ITR should be able to check that the `Map-Reply` messages received, following a `Map-Request` message, has not been altered on its way between the mapping system and the ITR. The answering ETR has to be authenticated to ensure that the latter is legitimate to answer to the `Map-Request`. Finally, the LISP-DDT mapping system also has to be secured.

All the security aspects of the data plane that just have been mentioned are discussed in more details in this chapter.

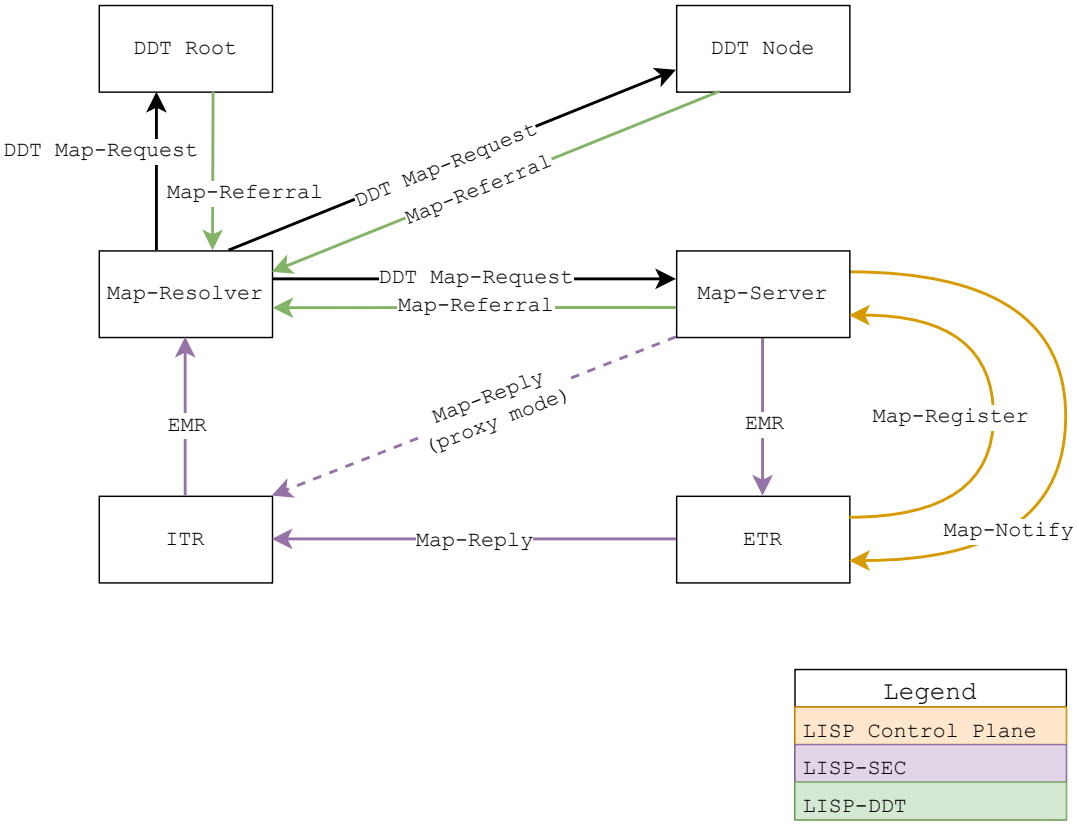

FIGURE 3.1: Summary of the possible messages exchanged between the different node types implied in the control plane. The colour of the arrays identifies how security is provided for each message. **EMR** means `Encapsulated Map-Request`.

3.1.2 Securing the control plane

In order to present the different aspects providing security in the control plane, let us review in a broad perspective the different interactions occurring in this plane. Six different types of devices participating in the mapping system can be listed :

- The ITR, asking for some mappings ;
- The ETR, registering EID-prefixes and replying to the `Map-Request` messages ;
- The Map-Resolver, at the interface of the mapping system, handling the requests in it ;
- The Map-Server, also at the interface of the mapping system, acknowledging and advertising the registered mappings ;
- The DDT node, composing the DDT tree structure ;
- The DDT root in particular, the root of the DDT tree.

It is important to note that a LISP node – a network device inside a LISP site associated with an EID – is not considered as a participant of the control plane. Although the LISP node indirectly triggers control plane events when LISP tunnels have to be established for its connectivity, this node is not aware of the LISP specification and is not inclined to handle LISP control messages.

Figure 3.1 summarizes the different possible messages – represented by arrows – that can be sent and received for every listed nodes. The usage and specification of those different messages are detailed in Chapter 2. Thanks to this figure, one can have a better insight about the interactions in the LISP control plane. This overall picture will enable us to present the different security aspects brought to the control messages and later in the chapter, enable us to highlight potential security issues on which to base the attack designed in Chapter 4.

The color code in Figure 3.1 represents, for each message, which document provides security on it. The document is either the LISP control plane specification [6], the LISP-DDT mapping system specification [7] or LISP-SEC [8], a protocol specifically tailored to bring security in the mapping lookup. Depending on the message, one needs security features such as, for instance, confidentiality or integrity of the message, authenticity of the peers, mechanisms to avoid replay attacks, and so forth. Let us describe each security provider listed above with particular emphasis on the security features that are provided by each of them.

3.1.3 LISP Control Plane security

A trust relationship has to exist between an ETR and a Map-Server for the registration process of EID-prefixes ; achieved by the use of the control messages represented by orange arrows in Figure 3.1. Indeed, the authentication of an ETR registering a certain EID-prefix is crucial. If it was not the case, any device could register for any EID-prefix and thus direct the LISP traffic to any location, enabling rogue and man-in-the-middle attacks. Such an attack is called EID-prefix hijacking.

Besides being able to identify the ETR sending the `Map-Register` message, the Map-Server must also check that this ETR is legitimate to register to the prefixes included in the message. Yet again, this verification is needed to avoid prefix hijacking, ensuring that the ETR does not try to intercept illegitimate LISP traffic.

Therefore, in order to countermeasure this issue, the `Map-Register` messages are signed by the ETR using one or multiple pre-shared secrets between the ETR and the Map-Server. The LISP specification assumes that those pre-shared secrets are already stated in the configuration of the different devices. The signature is included in the message in the **Authentication Data** field, as already stated in Section 2.1.3. This enables authentication and integrity of the message. In the same way and for the same reasons, the `Map-Notify` and `Map-Notify-Ack`[1] messages contains a signature in its **Authentication Data** field.

Nonetheless, a mechanism is still needed in order to avoid `Map-Register` replay attacks. The use of the **Nonce** field addresses this issue. For every `Map-Register` message sent by an ETR, the nonce is incremented. The Map-Server must keep track of the last nonce to ensure that one nonce is not used twice ; which would translate to a replayed message. As the content of the message – including the nonce – is signed, the value of the nonce cannot be altered by a potential attacker. Finally, the `Map-Notify` and `Map-Notify-Ack` messages have to set their **Nonce** field to the same nonce value as the related `Map-Register` message.

[1] The `Map-Notify-Ack` messages are not represented in Figure 3.1 for the sake of simplicity.

3.1.4 LISP-SEC

LISP-SEC [8] is the security extension for the LISP mapping data conveyed between the mapping system and the xTRs during a mapping lookup. This data is of course carried inside `Encapsulated Map-Request` and `Map-Reply` messages. In this manner, LISP-SEC provides, for those messages, origin authentication, integrity of the messages and anti-replay protection. It also provides protection against prefix overclaiming ; which will be discussed in more details in this current section. This is represented in Figure 3.1 with purple arrows.

Let us review the different operations happening during a mapping lookup with LISP-SEC enabled. In the scenario, an ITR wants to contact one of its known Map-Resolver in order to request an EID-prefix-to-RLOCs mapping. To this end, the ITR sends an `Encapsulated Map-Request` message to the Map-Resolver. The ITR also generates a one-time key (ITR-OTK) that will be used during the lookup for enabling the different security features. This key accompanies the `Encapsulated Map-Request` message and must be confidential. Therefore, its value is encrypted either by using a pre-shared secret between the ITR and the Map-Resolver or by enabling DTLS (Datagram Transport Layer Security) – the TLS equivalent for UDP packets [39].

Upon reception of the `Encapsulated Map-Request` message, the Map-Resolver decrypts the value of the ITR-OTK. As a DDT Map-Resolver, the device begins the iterative lookup. It is important to note that the LISP-SEC information is not transmitted to the intermediate DDT nodes but is only transmitted to the last element of the lookup ; the Map-Server.

In turn, the Map-Server receives a `DDT Map-Request` containing the LISP-SEC information – which includes the ITR-OTK. The Map-Server treats the request and find out which EID-prefix and ETR is relevant for the `DDT Map-Request` message. Before sending an `Encapsulated Map-Request` message to the ETR in question, the Map-Server first computes different security elements : the EID-prefix authorization and the Map-Server one-time key (MS-OTK). The prefix authorization is computed by applying an HMAC of the EID-Prefix with the ITR-OTK as the secret. Regarding the MS-OTK, its value is computed by applying a Key Derivation Function (KDF) on the ITR-OTK. Both EID-prefix authorization and MS-OTK are sent along the `Encapsulated Map-Request` message to the ETR. Let us mention that the MS-OTK is a confidential value ; it therefore has to be sent encrypted either by using a pre-shared secret between the ETR and the Map-Server or by using DTLS.

When the ETR receives the `Encapsulated Map-Request` message from the Map-Server, it first decrypts the MS-OTK. This secret will be used to generate the signature of the `Map-Reply` message. The EID-Prefix authorization must also be copied in the message. As one can notice, this authorization is a signature from the Map-Server of the EID-Prefix the ETR is responsible for. Because of the fact that the prefix is signed with a secret unknown to the ETR, the latter is not able to change the value of this prefix. If the ETR was able to do so, it could have tried to advertise to the ITR that it is responsible for a wider EID-Prefix, and thus eventually be able to intercept illegitimate traffic ; such an attack is called a prefix overclaim.

Upon reception of the `Map-Reply` message, the ITR is able to verify the prefix authorization – signed with the ITR-OTK, known to it – and the integrity of the message – signed with the MS-OTK. Indeed, the ITR is in turn able to compute the MS-OTK value, by using the same Key Derivation Function than the one used by the Map-Server on the ITR-OTK.

In conclusion, LISP-SEC ensures that the content of `Encapsulated Map-Request` and `Map-Reply` messages is not altered during its convey between the Mapping System and the xTRs. Moreover, it ensures that the EID-Prefix value is not altered by the ETR but is instead determined by the mapping system itself.

3.1.5 LISP-DDT security

Figure 3.1 shows that a trust relationship exists between the different participants of LISP-DDT. Indeed, a DDT Map-Resolver has to ensure that the received `Map--Referral` messages actually originate from the legitimate nodes and that their content has not been altered from the sender node to the Map-Resolver. In other words, origin authentication and message integrity has to be provided for the `Map-Referral` messages. Moreover, a Map-Resolver must trust all the different nodes participating in the XEID-Prefix delegation up to the Map-Server authoritative for it. As a matter of fact, an untrusted DDT node can potentially be a rogue node sending fake information about XEID-Prefixes.

The LISP-DDT security features listed above are provided using asymmetric cryptography. Each DDT node generates at least one key pair : a secret key and a public key. The secret key is used to sign the `Map-Referral` messages whereas the public one is used to verify the signatures. It is obvious that the different public keys have to be distributed amongst the different LISP-DDT participants needing them. Yet, a public key infrastructure is not needed in order to do so. Instead, LISP-DDT takes advantage of the fact that the DDT infrastructure consists of a tree. More precisely, every DDT node has to be aware of the different public keys of its child nodes. Hence, whenever sending a referral to one of its child node, the parent node includes the public keys of this child node in the `Map-Referral` message.

Thanks to this design, a Map-Resolver only has to keep track and trust the public keys of a DDT root. In this way, during an iterative lookup, the Map-Resolver will be able to establish an authentication chain from a configured trust anchor – the DDT root – until reaching the Map-Server. Indeed, the DDT root will send the public key of its needed child node and sign it. Because of this signature, the Map-Resolver can consider this public key as trusted. This mechanism is done for each referral until reaching the Map-Server. The discovered and trusted public keys can potentially be cached by the Map-Resolver for speeding up the upcoming lookups.

The signature of a `Map-Referral` message is situated at the end of the datagram, in the **Sig section** which was previously seen in Section 2.2.3. Figure 3.2 presents in details the content of this section. Amongst the different fields, one can notice the **Signature Expiration** and **Signature Inception** fields defining the time interval on which the signature is valid. In addition, the **Key Tag** field allows the signer to determine which key it used for the signature ; it allows one to own multiple keys.

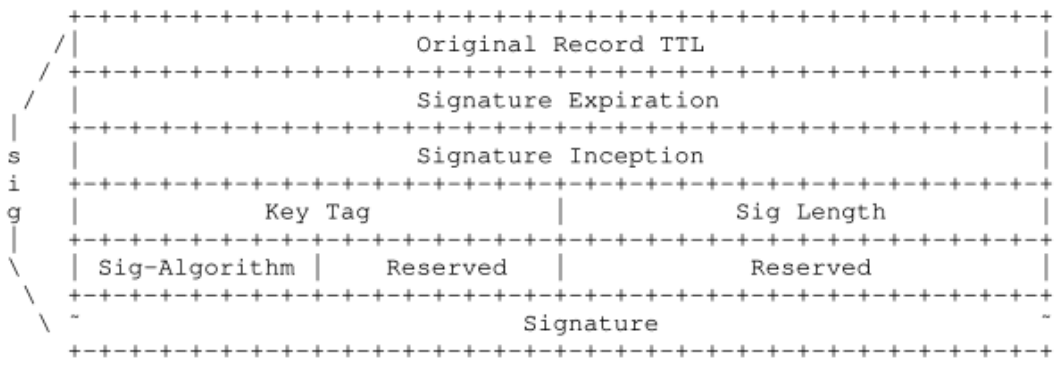

FIGURE 3.2: LISP-DDT Signature Section

Finally, the **Signature** field actually contains the signature of the entire `Map-Referral` record, using the cryptographic algorithm determined by the **Sig-Algorithm** field.

As previously stated, in addition to the signature, the `Map-Referral` message must contain the public keys of the referred child nodes. The public keys are included in the Record by using the special LISP Security LCAF Type.

The LISP-DDT security gives the possibility for a DDT node to revoke one of its key. In order to do so, a new record containing the public key is created and marked as revoked. The record must also be signed with the private key corresponding to the public key to revoke, in order to prove that the revocation indeed originates from the node owning this key.

Once revoked, the key should never be used and the node should advertise the fact that the key is revoked during all the life span of this key. This advertisement is done in a pull model : any DDT client is informed when querying the node. The parents of the node revoking a key should also be notified in order to stop advertising this key as valid.

As seen in previous section, LISP-SEC material is transmitted from a DDT Map-Resolver to a Map-Server. This includes the one-time key generated by an ITR along its `Map-Request` message. The confidentiality of this information is critical, therefore, this one-time key should be sent encrypted from the DDT Map-Resolver to the Map-Server. LISP-DDT is able to do so by the use of its public-key infrastructure. The one-time key can indeed be encrypted by using the Map-Server's DDT public key.

3.2 Threat analysis

A threat analysis for LISP has been developed by the LISP IETF working group [40]. The document in question principally defines a threat model tailored for the LISP protocol. The purpose of this section is to present this threat model in order to later use it as a canvas for highlighting possible security threats. From the different security threats that will potentially be found out, one in particular will be developed in more details in order to construct the denial-of-service attack detailed in Chapter 4.

3.2.1 Operation modes

The LISP Threat Analysis document provides multiple classifications for an attacker. The categories of attacker are reviewed in this present section. It is important to note that these categories are not mutually exclusive ; an attacker can be classified in any combination of these ones.

An **on-path attacker** (man-in-the-middle) is an entity that is able to intercept (and modify) messages between two entities. This can be achieved by either being located in the normal communication path or by deviating using any manner of means. An attacker that is not able to intercept such messages is thus called an **off-path attacker**.

Secondly, a **internal attacker** is an attacker whose attack is launched from inside a LISP site. Inversely, an attack that is launched outside of LISP sites is produced by an **external attacker**.

Next, a **live attacker** has to be active during all the duration of the attack. This means that the attack ends whenever the attacker stops its process. On the contrary, an attacker whose attack remains active after the action of the attacker is called a **time-shifted attacker**.

Finally, a **control-plane attacker** takes advantage of the LISP control plane functionalities – *e.g.* the mapping system – in order to build its attack. In turn, a **data-plane attacker** uses the LISP data plane functionalities for its attack. However, one can easily imagine an attack which takes advantage of both the data and control plane functionalities.

3.2.2 Threat categories

Besides the attacker categories, the LISP Threat Analysis document also provides categories for the different possible attacks. Once again, the categories are not mutually exclusive.

Here is the list of the different categories.

- A **replay attack** aims at retransmitting one or multiple packets without modifying their content.
- A **packet manipulation attack** means that an attacker intercepted a message, modified it and transmitted it to its final destination.
- A **packet interception and suppression attack** aims at intercepting a message and discarding it. Therefore, its final destination will not receive it.
- In a **spoofing attack**, the attacker pretends to be another node of the infrastructure. In order to do so, the attacker forges custom IP packets with another source address than its own. In the case of LISP, because of its use of encapsulation, the spoofing can either be applied to the inner header's address or the outer one.
- In a **rogue attack**, the attacker appears as a legitimate node. Contrary to a spoofing attack, the attacker does not need to fake its identity.

- A **denial-of-service attack** aims at making a specific network service to stop operating.
- In a **performance attack**, the attacker exploits computational resources of the victim so as to disrupt its proper functioning.
- An **intrusion attack** aims to gain access to a remote resource or information that is not supposed to be accessible.
- In an **amplification attack**, the attacker generates packets to a specific target that will itself generate a larger traffic than the one of the attacker.
- A **passive monitoring attack**, a technical attack about LISP traffic that is out of the scope of this work.

The different categories of attackers and attacks will allow us to have a better insight about the different possible attacks to design and the different security vulnerabilities to exploit.

3.3 Impersonating a node

Now, let us try to find security weaknesses to exploit in the LISP mapping system in order to build an attack taking advantage of it. In order to do so, let us review the different node types related to the LISP mapping system and analyse the possible breaches in the architecture. The different participants of the control plane were listed in Section 3.1.2 (and can be overviewed in Figure 3.1) : the ITR, the ETR, the Map-Resolver, the Map-Server and the DDT node – which can be a root in particular. Added to this, we can also try to discover possible direct or indirect interactions with the control plane as a LISP node or any kind of network device outside of any LISP site. In a general perspective, for each node type, we will try to answer this question : "If I impersonate a node, what do I have access to?".

3.3.1 ITR

As seen in Figure 3.1, the ITR should send packets to Map-Resolvers (`Encapsulated Map-Request` messages) and receive packets from either ETRs or Map-Servers (the `Map-Reply` messages). In addition, this node is at the interface of the data plane, it is a boundary router of a certain LISP site.

The process of impersonating an ITR in the sight of the underlying LISP site basically boils down to impersonating a router in a network. As this is not related to LISP in particular, this topic is not cover in the present work.

Considering the mapping lookup of the ITR, the latter device interacts with Map-Resolvers, ETRs and – indirectly or not – Map-Servers. The security of this interaction is provided by LISP-SEC for, among others, authenticating the different nodes. The LISP-SEC process was detailed sooner in this chapter. Yet, the one-time key that is used for enabling this security is generated by the ITR itself. Indeed, potentially any ITR node must be able to use a certain Map-Resolver in order to benefit from its service : the EID-prefix-to-RLOCs mapping lookup. Consequently, it is possible

to act as a rogue ITR ; whether LISP-SEC is enabled or not. Indeed, a device wanting to act as an ITR in the eyes of the control plane just need to send `Encapsulated Map-Request` messages.

In addition to this, it is also possible to act as an already existing ITR ; instead of doing a rogue attack, the node proceeds to a spoofing attack. This is achieved by forging `Encapsulated Map-Request` messages and changing the inner and outer source addresses accordingly so as to match the value that would have been generated by the ITR the attacker wants to spoof. In this scenario, the Map-Resolver receiving the spoofed `Encapsulated Map-Request` message will believe that it was sent by the victim ITR. Thereupon, the `Map-Reply` of this request will be sent to the ITR in question instead of being sent to the attacker. Of course, the victim ITR should certainly not treat the reply because the device should notice that no `Encapsulated Map-Request` message was sent by itself for this reply.

It is important to note that the spoofing of the outer IP header may not work in certain cases. Indeed, most of the routers will enable anti-spoofing policies and discard IP packets whose source address comes from an unexpected interface. However, anti-spoofing procedures for inner IP header addresses are not as straightforward as it would result to deep packet inspection.

In addition to the spoofing of an ITR presented here, let us notice that one can actually design a spoofing attack targeting not only an ITR but any network device. However, a device that is not related to the LISP control plane is not prone to have a network socket opened at the LISP Control Packets port (4342). In any case, the `Map-Reply` message packet will reach the network card of the victim, whether the packet is treated or not. This spoofing attack is at the source of the denial-of-service attack that is designed in Chapter 4.

3.3.2 ETR

Taking back Figure 3.1 as reference, one can notice that an ETR communicates with Map-Servers and ITRs. Depending on the procedure, the security is provided either by the control plane itself of by LISP-SEC.

Let us first analyse the `Map-Register` mechanism. As previously seen, the control messages exchanged between the Map-Servers and the ETRs are signed in order to provide authentication. This is done by using pre-shared secrets. Hence, it is not possible to impersonate a specific ETR as it would mean that we have information about those secrets.

Even with the fact that the registration process is secured – meaning that an attacker cannot fool a Map-Server by faking its identity –, let us analyse what an attacker could do if they happen to intercept the `Encapsulated Map-Request` messages from a Map-Server in place of a specific ETR. Upon reception of such a message, the ETR should decrypt the Map-Server one-time key (MS-OTK) in order to use it for signing the subsequent `Map-Reply` message. However, this key is either encrypted using a pre-shared secret between the Map-Server and the ETR – which is subject to happen as a pre-shared secret is anyway needed for the registration – or

by using DTLS. In both cases, the attacker will not be able to obtain the MS-OTK and will therefore not be able to sign the `Map-Reply` message in place of the legit ETR.

3.3.3 Map-Resolver

The Map-Resolver communicates with multiple nodes, as one can notice in Figure 3.1 : ITRs, DDT nodes – including DDT roots – and Map-Servers. Once again, let us analyse if an attack can impersonate a Map-Resolver in the eyes of one of the listed nodes.

A certain Map-Resolver receives `Encapsulated Map-Request` messages from the ITR that are configured to use this specific resolver for the mapping lookups. Thanks to LISP-SEC, a pre-shared secret or the use of DTLS is required for the transfer of the ITR one-time key. An attacker will therefore not be able to impersonate a certain Map-Resolver in the eyes of an ITR if LISP-SEC is enabled.

A Map-Resolver also communicates with DDT nodes for getting information in the mapping system. However, any node of the public Internet can potentially act as a DDT client – and thus send `DDT Map-Request` messages to DDT nodes – without problem. The `Map-Referral` messages answered does not depend on the identity of the DDT client ; the fact of impersonating another one seems pointless.

Finally, a Map-Resolver send `DDT Map-Request` messages to Map-Servers. Once again, the impersonation of another Map-Resolver does not seem relevant.

3.3.4 DDT root

Let us analyse what an attacker could do if they are able to impersonate a DDT root of the mapping system. Such a node communicates with DDT clients, which are most of the time Map-Resolvers. The latter Map-Resolvers are preconfigured so as to use a specific DDT root as the entry point for the iterative lookup. Along this configuration is stored the public keys of this mentioned DDT root, which is used to verify the signature included in the `Map-Referral` messages sent by this root. Because of the fact that an attacker will not know the relevant private keys, it is not possible to impersonate such a node, thanks to the security provided by LISP-DDT itself.

3.3.5 DDT node

The DDT root presented just before is a specific type of DDT node. Hence, the analysis of such a node is quite similar. Contrary to the DDT root, a Map-Resolver is not preconfigured with the public keys owned by the DDT node an attacker would try to impersonate. However, those public keys are communicated and signed by one of the DDT node's parent for which the Map-Resolver already has a trust relationship thanks to the authentication chain explained in Section 3.1.5. As a consequence, such a node cannot be impersonated.

3.3.6 Map-Server

A Map-Server – being at the interface of the control plane – communicates with several node types : Map-Resolvers, ETRs and possibly ITRs. What can happen if an attacker impersonates a Map-Server ?

With regard to the Map-Resolvers, the security is provided by LISP-DDT. Thus, the Map-Server owns public and private keys just as any DDT node ; which is used to sign the `Map-Referral` messages send to the Map-Resolvers. Additionally, the LISP-SEC information carried in the `DDT Map-Request` messages is encrypted thanks to these keys. Consequently, an usurper will not be able to sign such messages and thus impersonate a Map-Server in the eyes of the Map-Resolvers.

Taking the ETR into account, the prefix registration process is secured by the LISP control plane itself ; a pre-shared secret must exist between a Map-Server and an ETR. This scenario was already discussed when considering the impersonation of an ETR. It is therefore not possible for an attacker to act as a Map-Server in this case.

A Map-Server also send `Encapsulated Map-Request` messages to ETR whenever it receives a `DDT Map-Request` message from a Map-Resolver. However, an attacker wanting to impersonate a Map-Server and having received a `DDT Map-Request` message will need to decrypt the ITR one-time key so as to generate the relevant LISP-SEC information for the `Encapsulated Map-Request` message. In order to do so, the attacker would have to need the secret keys of the legit Map-Server. Hence, the attacker will once again not be able to impersonate a Map-Server in this regard.

Finally, a Map-Server which have prefixes registered in proxy mode is prone to directly send `Map-Reply` messages to ITRs without having to contact an ETR in the first place. In the case in point, the `Map-Reply` messages must still contain LISP-SEC information. However, we just saw that an usurper will not be able to generate such information.

In conclusion, the impersonation of a Map-Server is not possible in any possible aspect, if LISP-SEC is enabled.

3.3.7 LISP node

An attacker can be connected as a node in a LISP site, either in a legitimate manner or by gaining access to an existing LISP node. In such a case, there is no question of impersonation as the node is not part of the control plane. However, actions of this node can indirectly trigger events in the LISP control plane. This is indeed expected as the mapping system is used in order to have information allowing an ITR to encapsulate LISP traffic in the data plane. Still, attacks can be imagined in this manner. For example, one can try to overflow the ITR mapping cache by trying to connect with EIDs from many prefixes [41]. However, such kind of attacks goes beyond the scope of this work.

Chapter 4

Denial-of-service attack using the mapping system

4.1 Introduction

Taking back the different vulnerabilities of LISP highlighted in Section 3.3, let us design an attack. As we already have mentioned, any node is able to act as an ITR in the eyes of the mapping system – the Map-Resolver in particular. In addition to this, the spoofing of the addresses in the `Encapsulated Map-Request` messages sent to a Map-Resolver seems possible to do.

In response to an `Encapsulated Map-Request` message, the mapping system sends back a `Map-Reply` message containing a list of records ; the ETR RLOCs relevant to the queried EID prefix. This topic was approached in details when describing the LISP control plane in Section 2.1.3. Starting from this idea, one can notice that the list of records – and therefore the length of the `Map-Reply` packet – depends on the queried prefix and the topology of the deployed LISP infrastructure.

If the packet length of the `Map-Reply` message happens to be significantly higher than the length of the associated `Encapsulated Map-Request` message, the mapping system can be used as an amplification attack vector. Yet, the traffic generated by the mapping system has to be directed to a certain victim. This is possible by spoofing the address of the forged `Encapsulated Map-Request` messages sent by the attacker to the mapping system. If the attacker generates a significant traffic, the mapping system will send a greater one to the victim, in function of the amplification factor of the attack. The mapping system will therefore exhaust a certain part of the victim's bandwidth that can eventually provoke a denial-of-service.

Taking back the threat model presented in Section 3.2, the attack described in this chapter is a **denial-of-service attack** (and potentially a **performance attack**) that is based on a **spoofing attack** and an **amplification attack**.

In the case of the attack presented in this chapter, the attacker is an **off-path** one because no message between two entities is intercepted. Additionally, the attacker is categorised as **external** to a LISP site. Indeed, the attack relies on trying to act as an ITR in the eyes of the mapping system. The ITR, being at the edge of a LISP site, communicates with Map-Resolvers by using a RLOC address instead of an EID one. Finally, the author of the attack is a **control-plane** and a **live attacker** respectively because the attack exploits the control-plane as an amplification vector and because the denial-of-service is only operative while the attacker is sending traffic.

The attack described in this chapter is an example of the myriad of existing denial-of-service attacks by amplification. Indeed, multiple UDP protocols (such as SNMP, NTP, DNS, BitTorrent and multiple game protocols) are sensible to amplification attacks [42]. In our particular case, LISP is the exploited UDP protocol for the attack studied in this work. This attack is particularly comparable to the amplification attack exploiting the DNS infrastructure [43]. The latter attack describes the use of either DNS open resolvers or authoritative name servers as a way to amplify the traffic targeted to a victim in order to perform a denial of service. One or multiple attackers send spoofed DNS queries to the DNS infrastructure that will answer with significantly larger packets to the victim. Hence, this attack is comparable to the one presented in this work because it uses spoofing and amplification techniques. The main difference is in the amplification vector ; the presented attack uses a LISP-DDT infrastructure instead of a DNS one.

This chapter is divided into four sections. The first one describes the amplification factor, a metric that serves as a way to measure the severity of the attack. Moreover, this section will analyse the value of this metric in the LISP Beta Network, the project that deployed a worldwide LISP infrastructure on the public Internet. Next, the second section details step by step the modus operandi of the attack in a simulated LISP network. The configuration of all the nodes forming this network is also detailed in this second section. The third section, in turn, presents and analyses the results of the simulated attack. Finally, a discussion about the different solutions to adopt in order to mitigate this denial-of-service attack in the future is provided in the last section of this chapter.

4.2 Amplification Factor

As we just have seen, the main idea behind the attack is to send `Encapsulated Map-Request` messages to the LISP-DDT mapping system in order for it to answer with `Map-Reply` messages to the attacker. This interaction is effective if the reply packet length is bigger than the querying one. In this regard, let us define a metric on which to evaluate the significance of the attack: the **amplification factor** [44]. Considering a message size as the total length of the full packet – including the IP header and the Ethernet frame –, the amplification factor is defined as $A = \frac{\text{size(reply)}}{\text{size(request)}}$. The goal of the attack is indeed to maximize this metric. In that sense, if $A < 1$, the packet sent to the victim is smaller than the one sent by the attacker, the amplification is not favourable in that case. Additionally, if $A = 1$, the victim receives as much traffic as the sender is sending, there is very little reason to going through the mapping system for the attack instead of directly sending the traffic to the victim. Finally, if $A > 1$, the amplification is meaningful for the sake of the attack. Still, the goal of the attacker is to have the highest possible value for the amplification attack. Therefore, the attacker aims at sending `Encapsulated Map-Request` messages that are as small as possible while making the mapping system generate large `Map-Reply` messages.

Figure 4.1 shows the smallest `Encapsulated Map-Request` message that still makes the mapping system reply. The attack presented in this chapter focuses on control messages whose inner and outer headers are IPv4. The different fields of a typical `Encapsulated Map-Request` message were presented in Section 2.1.3 at page 8 when

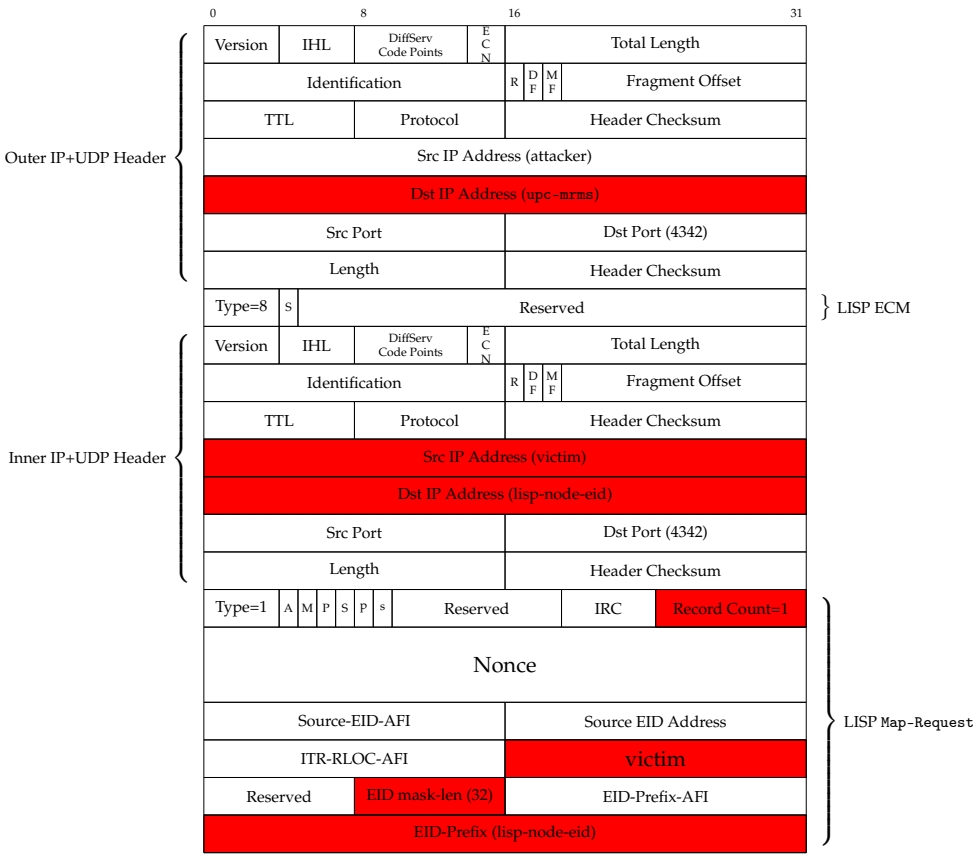

FIGURE 4.1: `Encapsulated Map-Request` message forged for the denial-of-service attack. The most noticeable fields are highlighted in red.

discussing the LISP control plane. In total, the message forged by the attacker is 102 bytes long, including the Ethernet frame (that is not visible in Figure 4.1). This message consists in requesting a certain EID mapping (**lisp-node-eid**) to a Map-Resolver (**upc-mrms**) while impersonating the victim node. One can also notice that the spoofing only occurs in the inner IP header as well as in the payload but not in the outer IP header. This particularity will be detailed later in this chapter, when performing the attack in a real-case scenario.

In response to this request, the mapping system will generate a `Map-Reply` message targeted to the victim. The size of this message – and thus the amplification factor – will depend on the EID prefix chosen in the first place by the attacker. More precisely, the length of the reply message depends on the number of ETR RLOCs registered in the mapping system for this specific EID prefix.

A negative `Map-Reply` message is simply, as a reminder, a `Map-Reply` message with no locator in its record. Such a packet has a total size of 70 bytes. However, each additional IPv4 RLOC in the record expands the packet by 12 bytes. Similarly, an IPv6 RLOC increases the packet size by 24 bytes. Figure 4.2 presents the size of a `Map-Reply` message depending on the number of RLOCs included in its record. It is important to point out that the amplification factor has to be bigger than one in order for the amplification to be meaningful. Therefore, the `Map-Reply` size should at least be 102 bytes long. In this way, a `Map-Reply` containing at least 3 IPv4 RLOCs, 2 IPv6

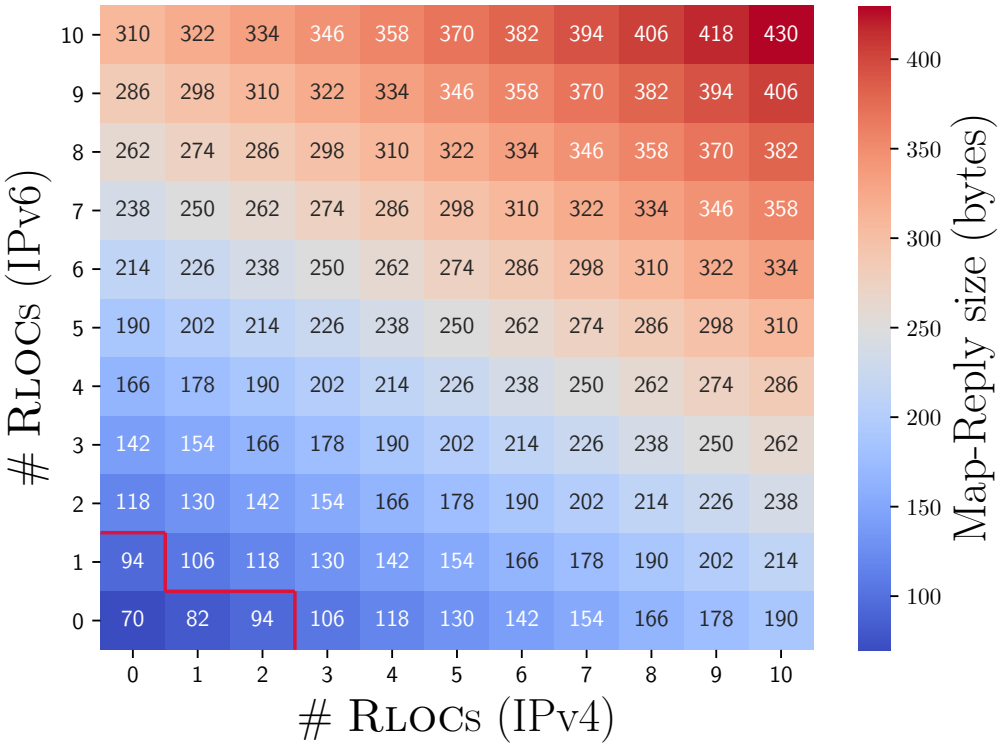

FIGURE 4.2: Size distribution of `Map-Reply` messages with respect to the number of RLOCs (IPv4 and IPv6). The red line separates messages that lead to an amplification factor < 1.

RLOCs or even one RLOC of each address type is sufficient. However, the attacker must maximize the amplification factor in order to have a significant amplification. For instance, the query of an EID prefix associated with four IPv4 RLOCs and four IPv6 ones – resulting in a reply of 214 bytes – can be used to approximatively double the traffic of the attacker.

Consequently, the goal of the attacker is to find an EID prefix to use for the attack on which there is as many registered ETR as possible. To go further, the attacker can also try to register by itself a certain EID-prefix in a way that a request for this prefix provokes a really large `Map-Reply` message. This situation is comparable to the DNS amplification attack on which an attacker inserts a large **TXT** record in the system [43]. However, in the case of LISP, the attacker will need to gain access – fraudulently or not – to a legit ETR in the network for inserting data in the mapping system as the registration process is authenticated.

It is important to mention that this attack can easily be extended to a **distributed denial-of-service** thanks to the fact that a single LISP can contain multiple ETRs. Indeed, the ETRs of the exploited LISP site are the devices that send the potentially large `Map-Reply` messages. The attacker only needs to send the spoofed `Encapsulated Map-Request` messages to the mapping system. The latter system will take care of distributing the multiple requests among the different ETRs that are currently reachable for the LISP site in use. This distribution is done according to the weight specified for each ETR in the locator set at registration. In conclusion, a LISP site containing a significant number of ETRs has multiple advantages regarding our attack. On the one hand, it gives a big amplification factor, leading to a more powerful attack. And on the other, the attack becomes distributed because multiple devices – the ETRs – will target the victim at the same time with large packets.

In order to prove the feasibility of the attack, let us implement one in a real-case scenario. In this regard, the attack will be performed in a simulated environment. However, the topology of this environment – especially the LISP-DDT mapping system – mimics the one of the LISP Beta Network [33], the worldwide LISP network deployed in the public Internet. This network uses LISP-DDT for its mapping system.

The first objective of the proof-of-concept is to have a good insight about the structure of the LISP Beta Network. This means that we have to inquire about the structure of its mapping system, the different Map-Servers and Map-Resolvers that exist and the different LISP sites that are registered including the number of ETR for each of them.

Let us mention that the LISP Beta Network ceased to provide its service since February 2020. Prior to this date, the now unavailable home page of the project gave information about the topology of the control plane. The EID space was divided into three parts according to geographical areas : US, EU and Asia. For each geographical area, two MRMSs (router combining the function of a Map-Resolver and a Map-Server) were present :

- `eqx-mrms` and `cisco-mrms` for the US ;
- `intouch-mrms` and `upc-mrmrs` for the EU ;

- `iij-mrms` and `sg-mrms` for Asia.

Therefore, the control plane interface consisted in those six devices.

The mapping system in use was LISP-DDT. As expected, the devices composing this system were organised in a tree. The leaves of this tree were the six MRMSs presented above. One DDT root (`ddt-root`) was present as well a two child nodes of this root : `ddt-tld1` and `ddt-tld2`. Each of theses child nodes were responsible of the whole XEID range and therefore of the six MRMSs.

Region	IPv4 EID Prefix
N/A	`132.227.120.176/28`
USA	`153.16.5.0/24`
	`153.16.10.0/24`
	`153.16.17.16/28`
	`153.16.23.240/28`
	`153.16.25.176/28`
	`153.16.25.192/28`
	`153.16.29.0/28`
	`153.16.30.96/28`
EU	`153.16.32.224/28`
	`153.16.36.0/24`
	`153.16.44.144/28`
	`153.16.58.208/28`
	`153.16.59.224/28`
Asia	`153.16.64.0/24`
	`153.16.67.0/24`
	`153.16.70.0/28`
	`153.16.70.64/27`
	`153.16.71.192/28`
N/A	`199.212.124.0/24`

TABLE 4.1: List of the IPv4 EID prefixes (each one corresponding to a LISP site) that are connected to the LISP Beta Network. This data was collected on November 4th, 2019.

In order to have information about the different LISP sites registered to the mapping system, we used the data provided by LISPmon. LISPmon is a monitoring platform that performed daily scan of the whole IPv4 EID space via `Encapsulated Map-Request` messages [45]. We collected the data on November 4th, 2019 which resulted in twenty LISP sites registered to the mapping system at that moment. The list of these LISP sites is available on Table 4.1. For the purpose of the attack, the number of IPv4 and IPv6 ETR RLOCs registered for each LISP site is an important metric as it is related to the amplification factor of the attack.

Figure 4.3 presents the CDF of the amplification factor for the twenty LISP sites. One can notice that, for more than 20% of the LISP sites, the amplification factor is bigger than 1. The LISP site provoking the mapping system to generate the longest `Map-Reply` message contains four IPv4 RLOCs, resulting in a packet size of 118 bytes. Hence, with the current mapping system state, the best amplification factor is equal

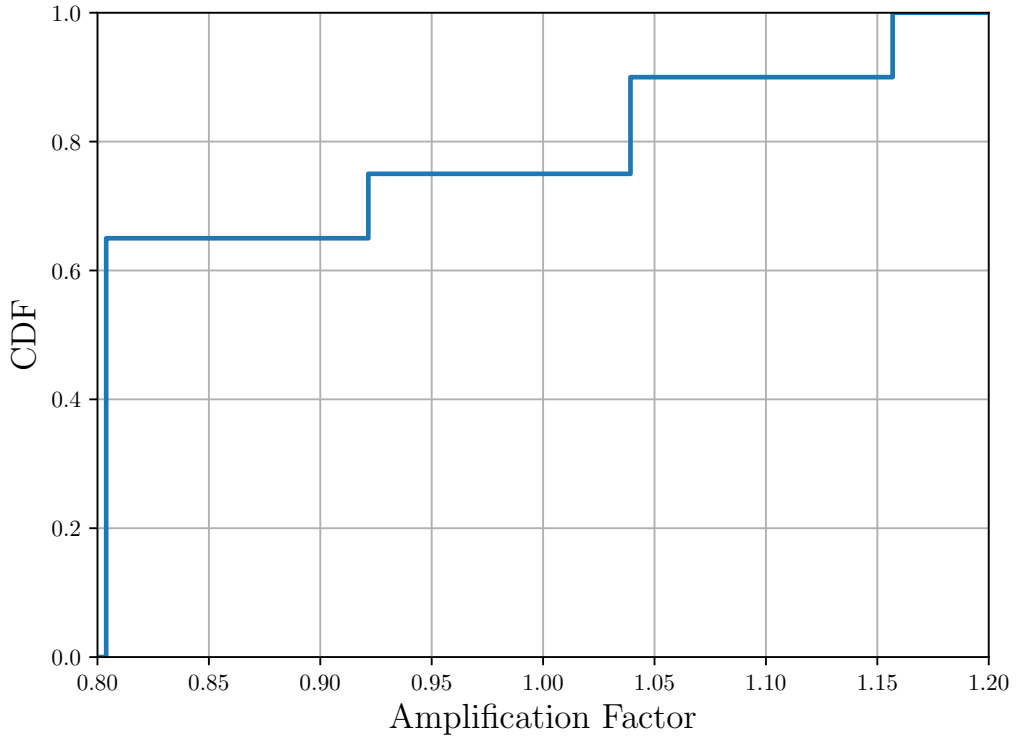

FIGURE 4.3: Amplification factor present in the LISP Beta Network.

to ~ 1.16. This value is relatively small but let us point out that the LISP Beta Network had a prototype-sized topology. Of course, this inconvenience does not change the fact that one attacker can potentially trick the mapping system by registering a long list of RLOCs if this attacker happens to have access to at least one ETR.

4.3 Description of the Attack

4.3.1 Logical Testbed

In order to run the actual attack, we reproduced the LISP Beta Network topology in GNS3[1] (Graphical Network Simulator-3), a network simulator. This software allows us to emulate the software of different network entities such as switches, routers and end hosts. All routers used in the following topology emulate a Cisco IOS C7200 image, a router operating system that is compatible with LISP and LISP-DDT. In turn, the end hosts are either GNU/Linux operating systems or simple VPCS (Virtual PC Simulator), a lightweight program simulating an operating system supporting basic networking functionalities.

Figure 4.4 shows the logical testbed topology that is used for the attack. The Internet core is represented by four different routers : `DFZ`, `US`, `EU` and `Asia`. With regard to the LISP-DDT mapping system, the simulated topology accurately mirrors the LISP Beta Network one. In this way, the three DDT routers and the six MRMS (Map-Resolver and Map-Server) that were listed in Section 4.2 are indeed present. The `victim` and `attacker` are respectively connected to the `EU` and `Asia` subnetworks as

[1] https://gns3.com/

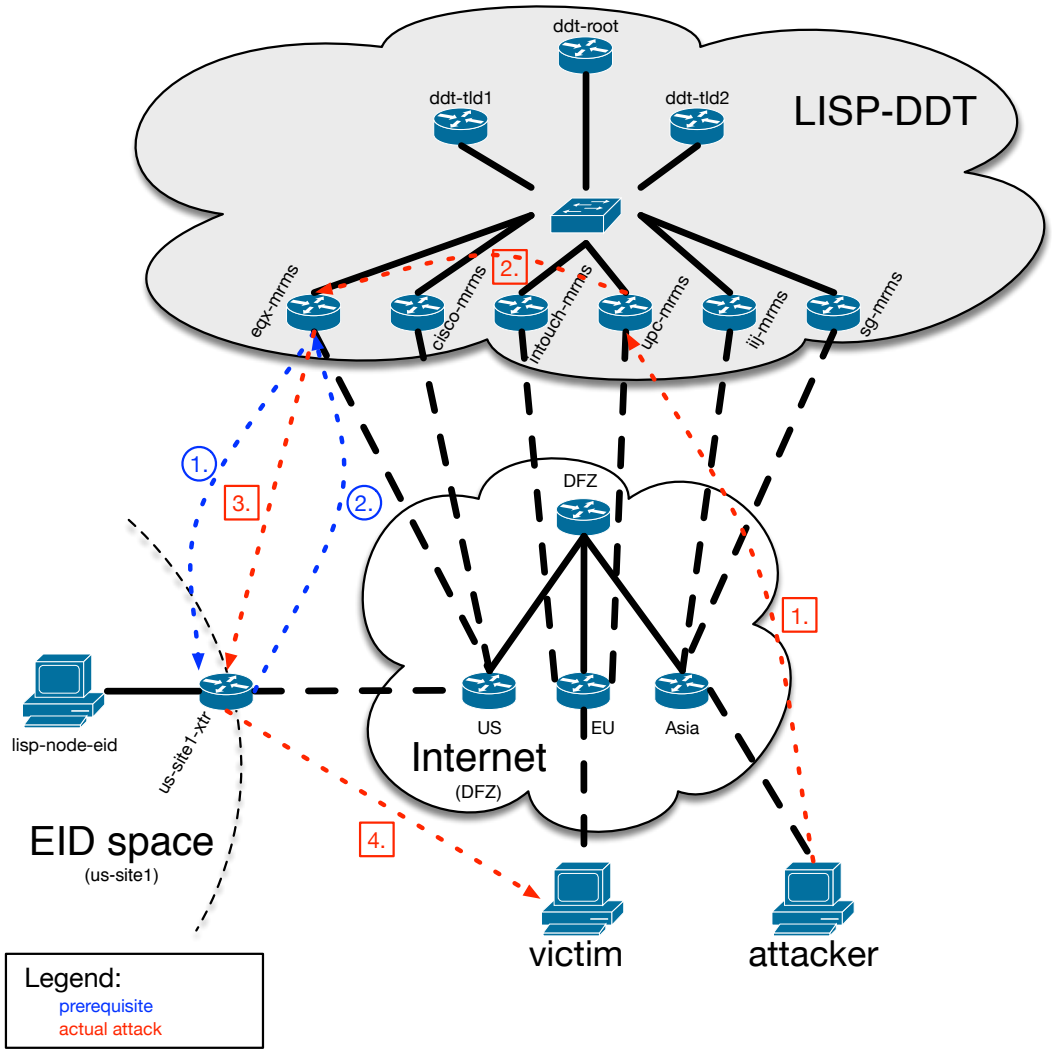

FIGURE 4.4: GNS3 logical testbed topology. The arrows represent the main steps for the attack.

non-LISP nodes. Finally, a LISP site, `us-site1`, is present in the simulated topology. This site is made of the `us-site1-xtr` xTR[2] and of `lisp-node-eid`, a simple LISP node.

Now that the logical testbed topology was outlined, let us review how the different devices of the network were configured. In particular, let us review the configuration of the different routers and end-hosts of the topology[3].

As previously stated, each router runs a Cisco IOS C7200 image. This operating system is specifically designed for routers and switches. Moreover, this particular version supports the LISP and LISP-DDT protocols. In that sense, a Cisco router is able to work as an ITR, an ETR, a Map-Resolver, a Map-Server and a DDT node.

[2] A LISP router being an ITR and an ETR at the same time

[3] All the configurations and scripts presented here are available at https://gitlab.com/m.gabriel/lisp-ms-ddos.

In order to design the DDT mapping system, multiple Cisco routers have to be configured : `ddt-root` as the DDT root of the tree, its two child DDT nodes `ddt-tld1` and `ddt-tld2` and the different Map-Resolver/Map-Server designed as `*-mrms`. All these devices are interconnected in a common virtual subnet.

Regarding `ddt-root`, this device is authoritative for the `0.0.0.0/0` IPv4 EID space as well as the `::/0` IPv6 EID space. This indeed means that this DDT node is responsible for the whole EID space. In addition to this, it delegates the `153.16.0.0/16` and `2610:D0::/32` EID prefixes to both `ddt-tld1` and `ddt-tld2`. Thus, both TLDs are responsible for the same EID prefixes but this design enables a load balancing of the different requests inside the mapping system.

As we just said, `ddt-tld1` and `ddt-tld2` are each responsible for the same prefixes. Hence, they both have the same DDT configuration. This configuration consists in delegating different parts of the EID-prefixes to the relevant Map-Servers. It is important to mention that the DDT node must state that the delegation points to a Map-Server. Indeed, the action code of the `Map-Referral` message sent by the DDT node depends on the nature of the referred child node : another DDT node or a Map-Server. This topic was detailed in Section 2.2.3. In that way,

- the `153.16.0.0/19` and `2610:D0:1000::/36` prefixes are delegated to both `eqx-mrms` and `cisco-mrms`;
- the `153.16.32.0/19` and `2610:D0:2000::/36` prefixes are delegated to both `intouch-mrms` and `upc-mrms`;
- the `153.16.64.0/19` and `2610:D0:3000::/36` prefixes are delegated to both `iij-mrms` and `sg-mrms`.

Regarding the different `*-mrms` nodes, it has to be configured in order to be able to communicate with the DDT mapping system as well as the different xTRs. As a DDT Map-Server, the XEID prefixes the device is responsible for is specifically stated. The prefix values for each MRMS are the same than the one configured above, in the delegation of the different TLDs. Additionally, those devices act as DDT clients because of their Map-Resolver nature. The address of the root DDT node is therefore present in their configuration. Finally, traditional LISP Map-Resolver and Map-Server features are enabled for those devices.

For the specific case of `eqx-mrms` and `cisco-mrms`, some configuration relating to the `us-site1` LISP site must exist, even before any registration from an ETR of this site happens. Indeed, a pre-shared secret between the Map-Server and the potential ETRs of this site has to be put in place, as already mentioned in Section 3.1.3 when discussing the security aspects of the LISP prefix registration process. In addition of the pre-shared secret of the `us-site1` LISP site, the Map-Server also keeps track of the potential EID prefixes this site can be responsible for. This indeed prevents an ETR to establish a prefix overclaiming attack. In this topology, the `us-site1` LISP site is responsible for the `153.16.0.0/24` and the `2610:D0:1000::/48` EID prefixes.

Now that the mapping system is set up in the GNS3 environment, let us see how the devices belonging to `us-site1` are configured : `us-site1-xtr` and `lisp-node-eid`. The `us-site1-xtr` router acts has an ITR as well as an ETR. Regarding its

ITR features, the configuration mentions which Map-Resolvers to contact for the mapping queries. The `eqx-mrms` and `cisco-mrms` devices are the Map-Resolvers in question. The `us-site1-xtr` router is also an ETR. In that sense, the configuration specifies which Map-Servers – `eqx-mrms` and `cisco-mrms` – to contact in order to register its LISP site on the mapping system. In addition, it includes the pre-shared secrets between the Map-Servers and itself. Of course, a locator set of ETRs has to be configured. This information is important as it is used during the attack in order to influence the size of the `Map-Reply` messages sent by this LISP site and thus the amplification factor. For each locator in this set, a priority and a weight can be tuned in order to enable traffic engineering and load balancing. It is important to mention that elements of the locator set must not necessarily really exist in this topology. In any case, the entry for this locator will be present in the different LISP control messages but the locator will be flagged as not reachable.

The `lisp-node-eid` device is also an element of the `us-site1` LISP site. It is actually the only LISP node of this site. Its function in the simulation is just to test the reachability of LISP packets and the good configuration of LISP in general. Practically, this LISP node is a device running a VPCS (Virtual PC Simulator). This allows us to attribute an EID address to the device (`153.16.1.2`, indeed belonging to the EID prefix the LISP site is responsible for) and send ping packets. The gateway of this device is the `us-site1-xtr` ITR.

At this stage, the mapping system as well as one example LISP site are configured. It is now time to link everything in a network representing the core internet. This network will also allow us to connect the attacker and victim devices, their configuration is discussed just after. The core internet, in the simulation, is composed of four routers : `US`, `EU`, `Asia` – each representing a geographical area of the Internet – and finally `DFZ` that links the three former between each other. Figure 4.4 summarises the structure of those core routers. The routers from the core network are configured so as to be able to forward packets whose addresses belong to the RLOC space. No packet having EID addresses should cross those routers. Let us mention that our simulated network does not run any BGP whatsoever because of its relative simplicity. Instead, the forwarding rules are configured statically. Let us also note that no anti-spoofing policy, such as reverse path forwarding, is configured in the different routers.

The purpose of the `attacker` node is to execute the script performing the attack. This script will forge multiple `Encapsulated Map-Request` messages and send them at a certain rate so as to perform a denial-of-service attack by amplification on the `victim` node. This script is written in Python and the `attacker` device runs a lightweight Debian distribution on which Python 3 is installed. The script executing the attack is based on a LISP looking glass developed by Tom Moraux for his Master Thesis [46]. Hence, most of the code executed by the `attacker` node belongs to the mentioned author. Nevertheless, his code has been modified so as to forge custom `Encapsulated Map-Request` messages at a certain rate. The specification of the forged `Encapsulated Map-Request` messages is described more precisely later in this section, when describing the process of the attack.

Finally, the `victim` node on which the attack is targeted is yet to be configured in the network. As a reminder, the amplification attack consists in making the mapping system send potentially long `Map-Reply` messages. Those kind of messages are meant to be sent to ITRs. In the case of this proof-of-concept, we assume that the victim node is not LISP-enabled, and is therefore not an ITR. Nevertheless, the `Map-Reply` messages will travel until reaching the victim device, still taking a considerable part of the bandwidth and thus, enabling a denial-of-service. Yet, in order to analyse the impact of the attack, a simple Python server listening at the LISP Control Packet UDP port (4342) is implemented on the `victim` node. Moreover, simple Bash and Python scripts were created in order to collect and process the different data relevant for analysing the results in Section 4.4.2. In this way, similarly to the `attacker` node, the `victim` device is lightweight Debian distribution that is able to interpret Python and Bash scripts[4].

4.3.2 Process of the Attack

The testbed network is now duly configured and running, this subsection describes the exact events happening during the attack and the possible configurations that can be changed in order to improve its impact. Figure 4.4 on page 46 gives a broad view of the attack ; the different steps are represented by dashed arrows.

The `attacker` node will take advantage of the fact that a LISP site containing a long locator set of ETRs is registered in the mapping system. The `us-site1` LISP site is an example of such. For the attack, the ETR of this site – `us-site1-xtr` has been configured such as to contain in its locator set four IPv4 RLOCs and four IPv6 RLOCs. One of the eight locators is the actual IP address of the ETR in question whereas the seven other locators does not really exist in the topology. Figure 4.2 states that such a configuration will result in 214 bytes-long `Map-Reply` messages. The amplification factor is hence a little bit greater than 2.

Before that the attack occurs, a registration process has to happen between the LISP site and the mapping system. In that sense, the `us-site1-xtr` ETR sends a `Map-Register` message to its configured Map-Servers : `eqx-mrms` and `cisco-mrms` (**prerequisite step 1** on Figure 4.4). In response, the Map-Servers each sends a `Map--Notify` message to the ETR (**prerequisite step 2**). This message exchange is done once in a while in order to keep the registration up to date.

Now that the LISP site that is used for the amplification attack is successfully registered in the mapping system, the attack in question can occur. The different steps of the attack are summarized on Figure 4.4.

- **Step 1:** The `attacker` node sends a continuous stream of `Encapsulated Map--Request` messages to a Map-Resolver, `upc-mrms` in this example.
- **Step 2:** This Map-Resolver uses either the DDT infrastructure or its cache in order to retrieve which Map-Server to send the request. In this example, the `Encapsulated Map-Request` messages are sent to the `eqx-mrms` Map-Server.

[4]The aforementioned scripts are freely accessible at https://gitlab.com/m.gabriel/lisp-ms-ddos.

- **Step 3:** Upon reception of the `Encapsulated Map-Request` message, the Map-Server sends the request to one of the ETRs responsible for it. In this scenario, the only available ETR is `us-site1-xtr`.
- **Step 4:** The ETR receives the `Encapsulated Map-Request` message and send to the `victim` node a `Map-Reply` message which is longer than the original message. The replies are indeed sent to the `victim` node because the `Encapsulated Map-Request` messages were spoofed ; the mapping system thought that the requests originated from the `victim` node. More on this later in this section.

In conclusion, the `victim` node is drowned by all the LISP packets received. If those messages take a sufficient amount of the bandwidth, this can result in a denial-of-service for the `victim` node. The `us-site1-xtr` can be configured so as to contain more entries in its locator set and thus increase the amplification factor. Another variant is to use the proxy mode of the Map-Server. In that configuration, the mapping system directly send the `Map-Reply` messages to the `victim` node without even going through an ETR.

The source address in the `Encapsulated Map-Request` message has to be spoofed in order to make the attack possible. If it was not the case, the `Map-Reply` message would be directed to the `attacker` node. Let us go back to the `Encapsulated Map-Request` message fields forged by the `attacker` in Figure 4.1. Three different fields in this message mention the origin of the message : the source IP address in the outer IP header, the source IP address in the inner IP header and finally, the ITR RLOC entry in the LISP `Map-Request` section (whose value is `victim` in the figure). The straightforward idea is to set the `victim` RLOC in the three listed fields. However, the source IP address in the outer IP header is not taken into account by the mapping system in order to later send the `Map-Reply` message. As a consequence, the `attacker` is able to leave this field intact, or set its value to any IP address, when sending its `Encapsulated Map-Request` messages. Thanks to this, the denial-of-service attack can occur while being sure that the messages originated by the `attacker` are not discarded in a router by a potential anti-spoofing policy. Such policy is anyway not present in the test environment but can possibly be present in a real case, in the public Internet.

Going further with the `Encapsulated Map-Request` message format, in Figure 4.1, one can see in red the most important fields that have been set by the attacker. The outer **Dst IP Address** points to the RLOC of a Map-Resolver – the entry point in the mapping system – whereas the inner **Dst IP Address** corresponds to the EID of a LISP node (`lisp-node-eid` for instance). This LISP node belongs to the LISP site on which a mapping is requested in this `Encapsulated Map-Request` message. Indeed, this information is the same than the requested **EID-Prefix**, with an **EID mask-len** equalling to 32, meaning that the requested prefix actually represents a single address.

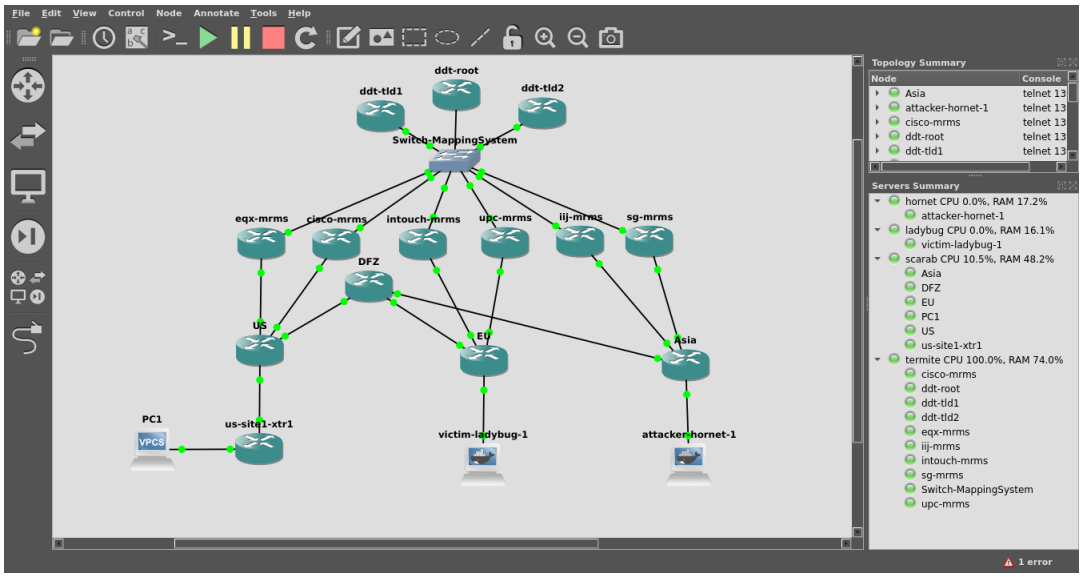

FIGURE 4.5: Screenshot of the GNS3 client graphical interface. The user has an easy access to information about the different GNS3 servers and about the emulated nodes of the topology.

4.4 Evaluation of the Attack

4.4.1 Physical Testbed

Now that we have the process of the attack in mind, let us analyse the impact when being run on our testbed topology. Before presenting and interpreting the results, let us get more insight about the GNS3 configuration that is used to simulate the logical topology and emulate the different network devices.

The GNS3 simulator is designed in a client and server architecture. In addition, multiple GNS3 servers can run concurrently on a same topology in order to balance the load of the simulator. This gives the possibility to scale the simulated topology without encountering constraints due to the specification of the device running the GNS3 server. Hence, each GNS3 server handles a specific part of the logical topology and one of them is configured as being the master GNS3 server that orchestrates the others. Figure 4.5 shows a screenshot of the GNS3 client user interface. The main part of the interface shows the current logical topology with the different nodes that were detailed in Section 4.3.1. In addition to this, the panel at the right side lists the different GNS3 servers dedicated to this topology as well as the different virtual nodes each of them handles. This panel also provide information about the CPU load and the amount of used memory of each GNS3 server.

In our current situation, four Linux machines from the Montefiore Institute networking lab (University of Liège) were dedicated to the simulation : `scarab` – the master server –, `hornet`, `termite` and `ladybug`. The servers are connected in two different subnets. The first subnet interconnects `scarab` and `hornet` whereas the other subnet contains `termite` and `ladybug`. Moreover, the bandwidth between both subnets is limited by a 4 Mbit/s link. Figure 4.6 shows the topology of the network lab.

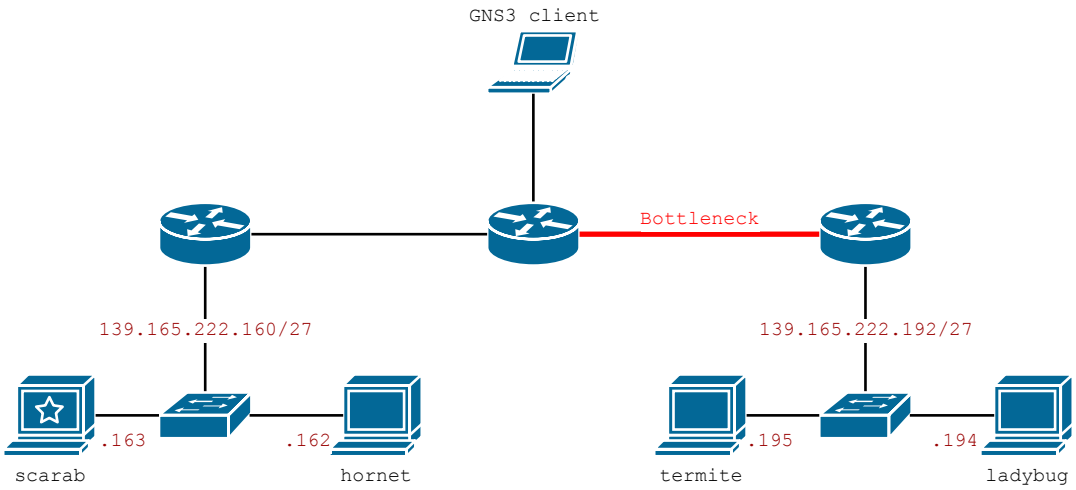

FIGURE 4.6: Topology of the actual network running the different GNS3 servers. The server represented by a star, `scarab`, is the master. A bottleneck is present between both subnets, limiting the bandwidth at 4 Mbit/s.

Note that the `GNS3 client` device does not participate in the simulation of the different devices in the virtual topology but is instead used to control and interact with them.

It is up to us to determine which device of the logical topology has to be emulated by which GNS3 server. We judged important to dedicate an entire server for the `attacker` node as well as the `victim` one. Thanks to this, the results probed in the `victim` emulated device are as little as possible influenced by the performances of the emulator. In that respect, the `hornet` GNS3 server emulates the `attacker` node whereas the `ladybug` server emulates the `victim` one. With regard to the two remaining GNS3 servers, `termite` emulates all the devices forming the mapping system (`ddt-root`, `ddt-tld1`, `ddt-tld2` and `*-mrms`) and finally, the `scarab` GNS3 server emulates everything else (`DFZ`, `US`, `EU`, `Asia`, `us-site1-xtr` and `lisp-node-eid`).

A better situation would have been to interconnect the four GNS3 servers on a physical network lacking the bottleneck link shown in Figure 4.6. However, the networking lab in use was also needed for other projects. It was therefore not possible to change its topology. During the simulation of the attack in the logical network, a significant traffic has to be generated between the different GNS3 servers, on the physical topology. Figure 4.7 represents the traffic generated between the different servers when they simulate the fact that the `attacker` node (emulated by `hornet`) generates a spoofed `Encapsulated Map-Request` message in order to attack the `victim` node (emulated by `ladybug`). As a reminded, the different steps of the attack are described in Section 4.3.2.

GNS3 is able to natively emulate the Cisco IOS operating system for each router as well as the VPCS for the `lisp-node-eid` device – which is configured so as to have an EID address belonging to the LISP site EID prefix. However, none of these emulated operating system is able to interpret Python scripts that are needed by the `victim` and the `attacker` nodes. Hopefully, GNS3 also allows one to import a Docker container as a node in the logical topology.

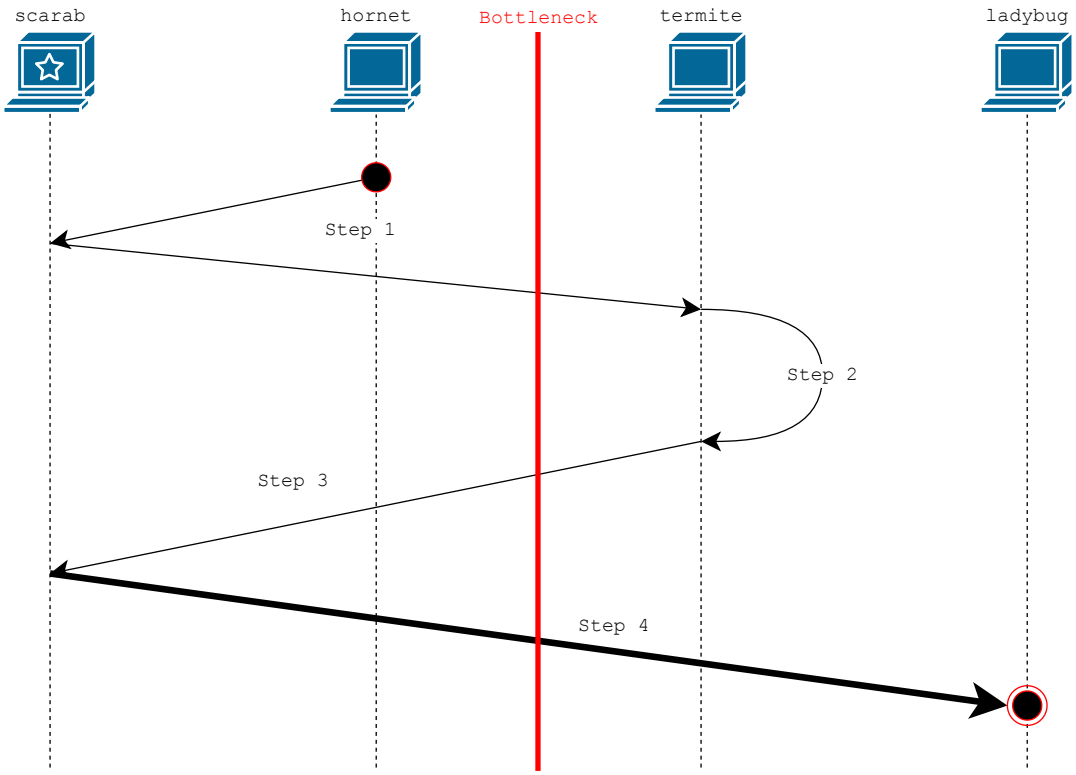

FIGURE 4.7: Analysis of the traffic generated in the physical topology during the simulation of the attack with regard to the bottleneck. The arrows represent the packets exchanged from a GNS3 server to another. The bold arrow is related to the amplified message directed towards the victim.

In this manner, the `victim` and `attacker` nodes are actually Docker containers running a lightweight Debian distribution that is able to execute Python code. Indeed, both victim and attacker run Python code in order to either generate traffic for the attack or collect data about its impact, as seen in Section 4.3.1.

4.4.2 Results

This section analyses the impact of the attack on the `victim` node. This impact depends on the intensity of the attack, characterized by the throughput of the attacker. Let us already mention that a simulated network, such as the present case with GNS3, creates performance limitations compared to an environment of real network devices. This limitation has to be taken into account when analysing the data probed during the attack.

As a reminder, the `victim` node in the topology runs a very simple UDP server at port 4342, the LISP Control Packet port. This is indeed the port number used by LISP Control Messages, `Map-Reply` messages in particular. This allows us to confirm the reception of the malicious data, product of the denial-of-service attack by amplification.

The simulations in which the different measures are recorded last 90 seconds. The `attacker` node send its traffic between second 30 and second 60 of the simulations. Still, 30 second periods prior and after the attack are recorded. Thanks to this, one can analyse the initial situation as well as the situation after the attack occurred.

Three different attack situations are presented, depending on the throughput of the `attacker`. In the first situation, the `attacker` sends 10 `Encapsulated Map-Request` messages per second. For the second and the last situations, this node respectively sends 100 and 1000 packets per second. Because of the limitation of the GNS3 servers hosting the topology as well as the network bottleneck present between the different servers, attacks with larger intensities gave no relevant results : the simulation was not able to handle that much traffic.

Figure 4.8 shows the amount of data received by the network card of the `victim` node for each second of the simulation. The two vertical dashed lines respectively represent the beginning and the end of the actual attack, period of time where the `attacker` sends traffic. Prior to the attack, one can see that hardly no network packets are received by the node. However, for the three situations, the node begins to receive a significant traffic at the moment when the attack begins. Moreover, as soon as the attack ends, the situation is back to normal. This proves that the designed denial-of-service attack is **live** (using the terminology presented in Section 3.2), it only has an impact whenever the attacker is active.

In the situation where the `attacker` node sends 100 packets per second ($\sim 10\,\mathrm{kB/s}$), the `victim` node receives a little more than $20\,\mathrm{kB/s}$ of data. This is indeed expected as the current amplification factor equals $214/102 \simeq 2.098$. A similar reasoning can be derived for the 10 packets per second situation.

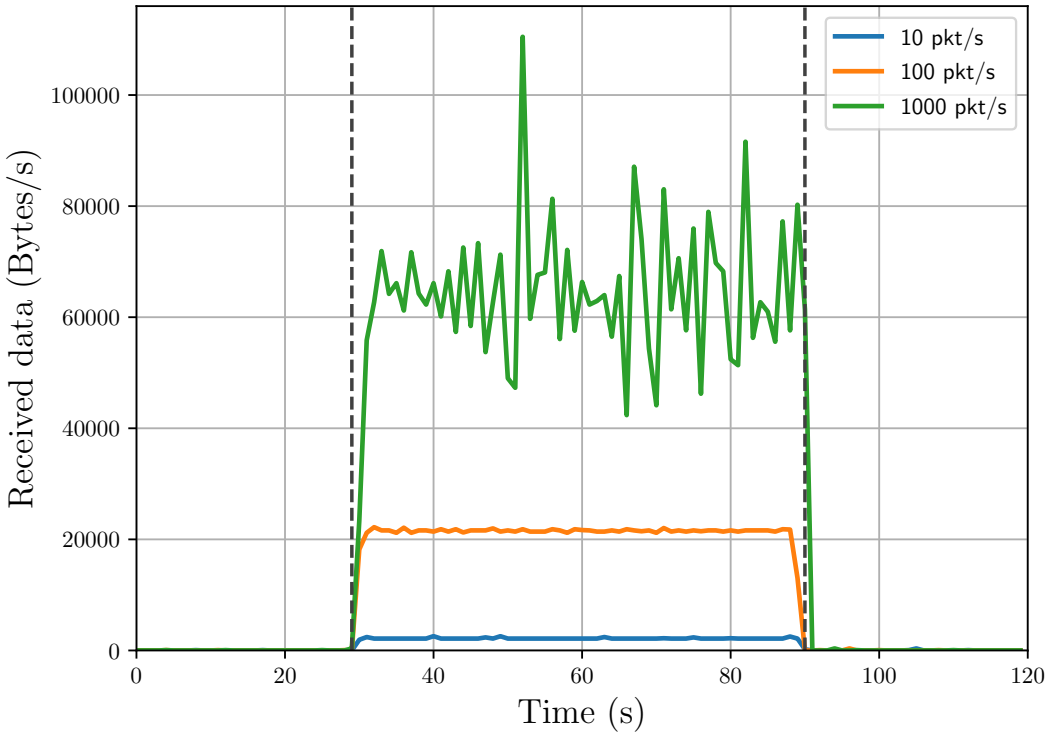

FIGURE 4.8: Received data on `victim` node.

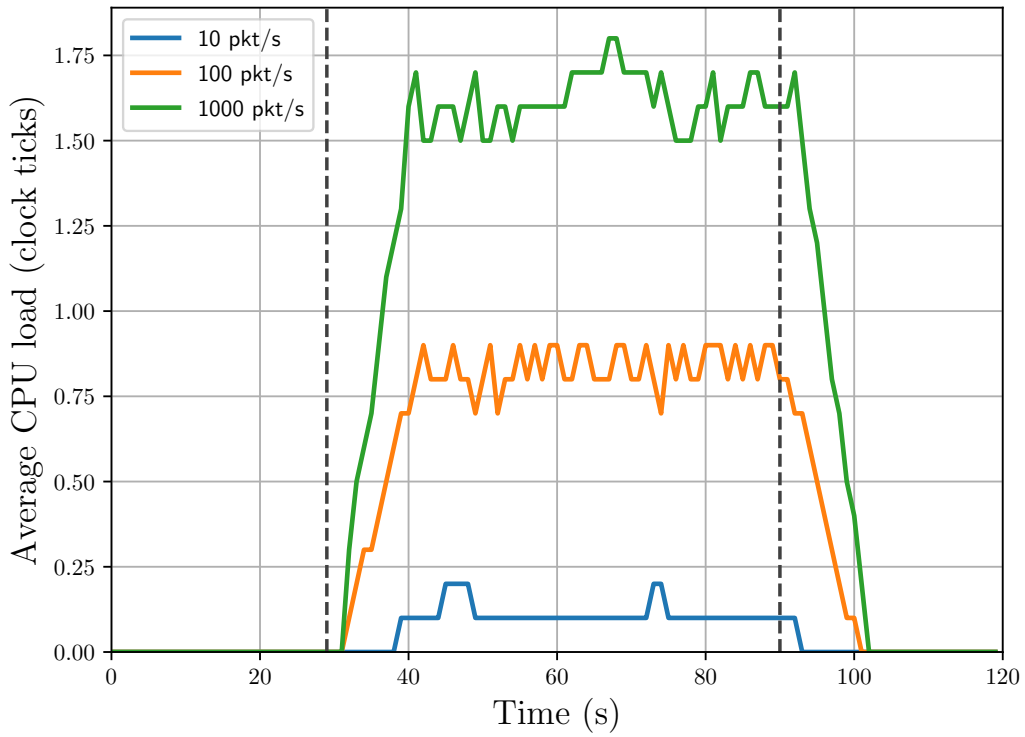

FIGURE 4.9: CPU overload on `victim` node.

In the case of the 1000 packet per second attack, the received data rate is expected to equal $\sim 200\,\mathrm{kB/s}$. However, on the graph, this value seems to fluctuate around the $65\,\mathrm{kB/s}$ value. This is the result of the fact that a significant number of packets were lost during their trip inside the simulated topology. As a matter of fact, when running this simulation, the GNS3 client reported a heavy usage, in term of CPU and RAM, of the different GNS3 servers managing the simulation. The GNS3 client interface indeed allows the user to view this information, on the right-side panel visible in Figure 4.5, page 51.

Moreover, the topology is distributed between the four GNS3 servers in a way that, for each `Encapsulated Map-Request` message sent by the `attacker` node, this message crosses twice the bottleneck link (Figure 4.7). In addition, the `Map-Reply` message – possibly really large – also has to cross this link. Taking into account that the maximum bandwidth in this link is $4\,\mathrm{Mbit/s} = 500\,\mathrm{kB/s}$, let us analyse the maximum possible throughput of the `attacker` node ($\mathrm{rate}_{\mathrm{request}}$) in the case where the amplification factor equals 214/102.

$$\begin{aligned} 500\,\mathrm{kB/s} &= 2 \cdot \mathrm{rate}_{\mathrm{request}} + \mathrm{rate}_{\mathrm{reply}} \\ &= 2 \cdot \mathrm{rate}_{\mathrm{request}} + \frac{214}{102} \cdot \mathrm{rate}_{\mathrm{request}} \\ &= \frac{418}{102} \cdot \mathrm{rate}_{\mathrm{request}} \end{aligned} \tag{4.1}$$

Following Equation 4.1, one can estimate that

$$\mathrm{rate}_{\mathrm{request}} = \frac{102}{418} \cdot 500\,\mathrm{kB/s} \simeq 122\,\mathrm{kB/s} \tag{4.2}$$

This means that, because of the fact that a `Encapsulated Map-Request` message is 102 bytes long, the `attacker` node can theoretically send 1196 ($= 122 \cdot 10^3/102$) packets per second at maximum. Hence, even if the GNS3 servers were able to treat that much information – which is not the case considering the multiple packet lost at that rate –, the topology of the networking lab would not have allowed us to simulate more traffic.

Figure 4.9 shows the CPU load of the `victim` node during the different simulations. In terms of being specific, this graph represents the average CPU load – over a time window of the past 10 seconds – of the Python server than runs on the `victim` node. The load caused by the other processes of the operating system are therefore not taken into account here. One can see that this process is idle prior to the attack. However, as soon as the attack is launched, the process generates a significant CPU load. The intensity of this load depends on the throughput of the `attacker` node. Finally, once the attack ends, the average CPU load gradually decreases until reaching a state similar to the initial situation. Thanks to this graph, one can conclude that the attack can potentially have an impact on the performance of the victim device (**performance attack** using the terminology of Section 3.2) in addition to the fact that the bandwidth can possibly be overloaded (**denial-of-service attack**).

4.5 Mitigation

As we have seen in this chapter, a potential large-scale deployment of LISP on the public Internet allows us to easily implement a denial-of-service attack on any network device thanks to the LISP-DDT mapping system. The attack that was described in this chapter is basically a combination of a spoofing attack and an amplification attack. Different techniques can be thought of in order to mitigate the impact of this attack. We can take inspiration from mitigation techniques already existing against denial-of-service attacks by amplification, broadly speaking. Furthermore, one can also imagine potential modifications on LISP and LISP-DDT in order to prevent one to use the mapping system as a vector for such an attack.

Regarding the amplification attack, a possible solution could be to prevent the amplification factor to be far too big. A straightforward solution would be to limit the number of ETRs that are registered in the mapping system for a certain LISP site. In addition, another possibility is to prevent the locator set of the `Map-Reply` message to contain unreachable locators. However, both solutions presented here will affect the usability of the LISP control plane as it is today. Because of the fact that LISP is not yet broadly implemented on the public Internet, it is hard to evaluate if such kind of limitations are constraining for the proper functioning of LISP. For instance, it is challenging to figure out the maximum reasonable number of ETRs a LISP site could need on a network as wide as the Internet. In any case, this limitation will prevent a LISP site to scale. This is therefore not the best practice to adopt.

The attack presented in this work remains a denial-of-service one. Its only novelty is the use of the LISP control plane as an amplification vector. Hence, any already existing technique designed to prevent this kind of attack can be set up in the network. For instance, one can detect denial-of-service attacks using deep learning models and convolutional neural networks in particular [47].

Another, more straightforward, solution is to include a rate limiter in the mapping system. One can imagine that the rate limitation could either be done according to the source address of an `Encapsulated Map-Request` message or the requested EID prefix in it. Such information inside the message could be use to segregate the traffic into different flows in order to potentially discard malicious ones (*e.g.* the one performing a denial-of-service attack). Hence, all Map-Resolvers on which the rate limiter is enabled should store statistics regarding the traffic it receives, depending on the different flows. This information can efficiently be stored in a hash-table or even more sophisticated data structures such as counting bloom filters [48] or count-min sketches [49], for example. Both mentioned data structures allow one to quickly check if the number of packets of a flow is smaller than a certain threshold.

If the requested EID prefix of the `Encapsulated Map-Request` message is the criterion that is used to determine the flows on which to limit the rate, multiple concerns has to be thought of. First, the attacker can easily change this information from a message to another when performing a denial-of-service using a certain LISP site. Indeed, this LISP site registered a certain EID prefix it is authoritative for in the mapping system. However, this does not prevent the attacker to request a more precise EID prefix which is a subset of the registered one. As a matter of fact, the attacker can request a certain EID prefix with a full mask length, which is equivalent to request

a single EID address included in the set of EIDs the LISP site is responsible for. For instance, let us imagine that the LISP site registered the `153.16.1.0/24` EID prefix. Thus, this prefix contains $2^8 = 256$ different EID addresses (with a mask length of 32). Even more, the attacker can request prefixes whose mask length is between 24 and 32 for this example, doubling the possible prefix values to request. However, the rate limiter can possibly notice that all those prefixes belong to the same LISP site and therefore, classify all these requests in the same flow.

The second concern with the idea of rate-limiting the `Encapsulated Map-Request` messages based on the requested EID prefix is that it can enable other kind of denial-of-service attack. Indeed, one can imagine that an attacker send multiple `Encapsulated Map-Request` messages with a certain prefix to a Map-Resolver until reaching the threshold for this prefix in order to prevent other ITRs to perform legitimate `Encapsulated Map-Request` messages for the mentioned EID prefix. Because of this, an attacker can ensure that an entire LISP site becomes unreachable as the ITRs of other sites will not be able to find out its location via the mapping system and therefore, they will not be able to ensure the LISP encapsulation of packets sent to this site.

The other criterion on which to classify the different flows is to use the source address of the `Encapsulated Map-Request` messages. This source address can either be the one of the outer or the inner IP header. For the case of the outer source address, this value can be spoofed with any value. Indeed, during the proof-of-concept attack, the outer source address was left untouched whereas the entire `Encapsulated Map-Request` message was still spoofed so as to target the victim. The value of this field was the address of the attacker node but the latter can forge packets in a way that this value is set at random. Hence, all the `Encapsulated Map-Request` messages sent by the attacker would be considered as belonging to different flows and no rate limitation would apply.

With regard to the inner source address, the rate limiter would be effective for preventing the attempt of the studied denial-of-service attack. Indeed, this field must contain the IP address of the victim node. If the rate is limited, the severity of the attack is strongly reduced, obviously. However, in the same vein as above, the rate limiter can be used so as to prevent a legit ITR to perform mapping queries and hence prevent it to encapsulate LISP traffic to the desired destination.

The denial-of-service attack presented in this work relies on spoofing of `Encapsulated Map-Request` messages. A typical anti-spoofing technique is the ingress traffic filtering [50]. This prevents an IP packet whose source address is spoofed to enter the core of the Internet, the packet is instead discarded by the routers of the Internet Service Provider (ISP) on which the attacker device is connected. Although this technique is straightforward to implement and is known for several decades, spoofing is still permitted by lots of ISPs [51].

Besides, the spoofing performed by the attacker occurs on the inner IP header, which is considered as the payload of the IP packet for a router. In that sense, deep packet inspection features on routers between the attacker and the control plane should be implemented in order to check that the inner source address is legitimate. To further complicate matters, the inner source address can be an EID address. A

router performing inspection on this value should be able to detect if this EID address is legit, which is far more challenging to do than for RLOC addresses.

Nevertheless, Map-Resolvers can possibly detect inconsistencies between the outer and the inner addresses. For instance, this kind of device can verify that the source EID address in the inner header is indeed in accordance with the source RLOC address in the outer header. Combined with traditional anti-spoofing techniques, this could fully prevent spoofing to happen.

Another potential solution to prevent the studied attack is the use of a certain kind of authentication between the querying ITR and the Map-Resolver. A weak authentication solution similar to DNS cookies [52] can be implemented, which is used to prevent off-path amplification attacks. The basic idea is to include in the `Encapsulated Map-Request` message a client cookie as well as a server cookie in order to authenticate each other. The client cookie is computed by the ITR and depends on the ITR IP address, the Map-Resolver IP address and a secret only known to the ITR. In turn, the server cookie is computed by the Map-Resolver and depends on the ITR IP address, the client cookie and a secret only known to the Map-Resolver. This is designed in a way that the server does not need to keep the cookies in memory for each client since they can be computed at will. Prior to a request, the ITR has to ask for the server cookie. In the case of a request spoofed by the attacker, the response will be directed to the victim node. Thus, the attacker will never we able to obtain the server cookie needed for the following communication. For the case of the LISP mapping system, a solution could be to force cookies to be used in order to receive possibly long `Map-Reply` messages. Hence, for an `Encapsulated Map-Request` message without cookies, the mapping system could either choose to answer with a limited-length `Map-Reply` message, with a "no-cookie" error or even to discard the request completely.

When LISP-SEC is enabled, the ITR one-time key has to be sent encrypted to the Map-Resolver, as already discussed in Section 3.1.4. The encryption is done by either using a pre-shared secret between the ITR and the Map-Resolver or by using DTLS. Both cases could provide a secure authentication if a signature of the `Encapsulated Map-Request` message is included in the latter. If the policy of the mapping system is to force the use of LISP-SEC with authentication – or at least strongly encourage it–, spoofed `Encapsulated Map-Request` messages will not be possible anymore. Therefore, this solution can prevent the denial-of-service attack studied in this work to happen while relying on LISP-SEC, an already developed technology.

4.6 Conclusion

As we have seen in this chapter, a LISP architecture deployed on a network – and in particular a LISP-DDT mapping system – can be used in order to perform a denial-of-service attack by amplification. This is due to the fact that the mapping system is subject to reply with messages that are way longer than the messages sent. The replies can easily be redirected towards a victim node on the network in order to consume a significant part of its bandwidth. Afterwards, we analysed the severity of such an attack if it occurred on the LISP Beta Network, the now unavailable deployment of LISP on the public Internet.

As a proof-of-concept, the presented denial-of-service attack was performed in a contained virtual environment. The results of this battery of tests was analysed in order to highlight the risk of this kind of attack. We have taken care of analysing this data while taking into account the possible limitations of a simulated network that would not apply in a real-case scenario.

Finally, a discussion about the different way of mitigating this attack in a future large-scale deployment of LISP has been provided.

4.7 Ethical considerations

This chapter detailed the process of performing a (distributed) denial-of-service attack by exploiting the LISP-DDT mapping system. The presented work focused on the theoretical aspect of such an attack and possible mitigation techniques. We did not make any attempt to perform this attack in a real situation, including the LISP Beta Network. All experiments were done in an isolated environment by simulating a virtual network in GNS3.

However, this work could be used by a person with bad intentions in order to perform this attack in a real network. Although this situation is possible, the risks seem quite limited as for now. Indeed, the LISP Beta Network was the only globally deployed LISP architecture on the public Internet and it ceased operation since February 2020.

Instead, this work must be considered as a way to highlight the security vulnerability existing in the current specification of the LISP protocol, allowing a DoS attack to be set up. We hope to draw the LISP IETF Working Group attention to this issue in order to ensure that this vulnerability is mitigated by the time LISP is widely deployed on the public Internet.

Chapter 5

Conclusion

The routing system of the Internet is subject to scalability concerns. Indeed, the routing tables of the core routers grows more and more. This seems to be mainly due to different factors such as multi-homing, traffic engineering or even the demand of provider-independent IP addresses by end-sites. The Locator/Identifier Separation Protocol (LISP) addresses this issue by applying a certain separation paradigm on the address space. This aims at splitting the address space semantics into its two roles : location and identification. In that sense, LISP introduces a locator space (RLOC) – which is globally routable – and an identifier space (EID) – which is in contrast only locally routable. Hence, a device inside a LISP site is identified using its EID. However, the EID address has no meaning in the core network, the RLOC is therefore used and a tunnel between LISP sites is established. Thanks to this design choice, the RLOC addresses can be defined so as to reduce the size of the routing tables of the core routers whereas the EID addresses management is left to the responsibility of the LISP sites.

This address space separation requires a way to map elements from the EID space to the RLOC space. In that regard, a mapping system – such as LISP-DDT – is used to contain and retrieve this information. By means of this, a LISP site can thus request the mapping system in order to retrieve the needed mapping information so as to establish the data tunnelings. The different requests are done by mean of control plane messages, notably the `Encapsulated Map-Request` and `Map-Reply` messages.

In this Master thesis, the security aspects of LISP and LISP-DDT are analysed in order to find out potential security vulnerabilities on which to base an attack. This analysis lead to the discovery of a possible denial-of-service attack by amplification exploiting the mapping lookup mechanism. The vulnerabilities at the source of the attack are the ability to spoof `Encapsulated Map-Request` messages as well as the amplification opportunity of the mapping lookup. To be more precise, an `Encapsulated Map-Request` message sent to the mapping system could provoke it to return a longer `Map-Reply` message to the victim.

A GNS3 simulated testbed environment has also been set up for the purpose of performing the studied attack. The severity of the attack is defined by the amplification factor, which is the ratio between the reply and the request lengths. For this proof-of-concept, in order to mimic a relatively small topology, we chose to perform the attack with a small amplification factor. Still, results demonstrated that this attack consumes both network bandwidth and CPU load at the victim's side. The feasibility of this attack was therefore proven, both theoretically and experimentally.

Some mitigation techniques have been suggested in this work. One can for example choose to prevent the amplification by limiting the size of the responses or by strongly limiting the rate of the requests. Another idea would be to set up techniques that detect spoofed messages and prevent them to be treated by the mapping system.

In any case, we hope that this work could also serve as an alert for the LISP IETF Working Group. Also, we could possibly hope that a solution to prevent such an attack to happen in a future wide-scale deployment of LISP in the public Internet will be set up. This current work could also be an entry point to future works developing the different mitigation techniques that could be adopted. Finally, this document did not seek to be exhaustive about the security vulnerabilities regarding LISP and its mapping system. In that aspect, it leaves the door open to future finding about other vulnerabilities or attacks.

Bibliography

[1] D. Saucez, L. Iannone, O. Bonaventure, and D. Farinacci. Designing a deployable Internet, the Locator/IDentifier separation protocol. *IEEE Internet Computing*, 16(6):14–21, November/December 2012.

[2] M. Hoefling, M. Menth, and M. Hartmann. A Survey of Mapping Systems for Locator/Identifier Split Internet Routing. *IEEE Communications Surveys & Tutorials*, 15(4):1842–1858, January 2013.

[3] G. Huston. BGP in 2017, January 2018. URL https://www.potaroo.net.

[4] D. Meyer, L. Zhang, and K. Fall. Report from the IAB Workshop on Routing and Addressing. RFC 4984, Internet Engineering Task Force, September 2008.

[5] D. Farinacci, V. Fuller, D. Meyer, D. Lewis, and A. Cabellos-Aparicio. The Locator/ID Separation Protocol (LISP). Internet Draft (Work in Progress) draft-ietf-lisp-rfc6830bis-32, Internet Engineering Task Force, March 2020.

[6] D. Farinacci, F. Maino, V. Fuller, and A. Cabellos-Aparicio. Locator/ID Separation Protocol (LISP) Control-Plane. Internet Draft (Work in Progress) draft-ietf-lisp-rfc6833bis-27, Internet Engineering Task Force, January 2020.

[7] V. Fuller, D. Lewis, V. Ermagan, A. Jain, and A. Smirnov. Locator/ID Separation Protocol Delegated Database Tree (LISP-DDT). RFC 8111, Internet Engineering Task Force, May 2017.

[8] F. Maino, V. Ermagan, A. Cabellos-Aparicio, and D. Saucez. LISP-Security (LISP-SEC). Internet Draft (Work in Progress) draft-ietf-lisp-sec-20, Internet Engineering Task Force, January 2020.

[9] J. Beeharry and B. Nowbutsing. Forecasting IPv4 exhaustion and IPv6 migration. In *2016 IEEE International Conference on Emerging Technologies and Innovative Business Practices for the Transformation of Societies (EmergiTech)*, pages 1–5. IEEE, August 2016.

[10] L. Cittadini, W. Muhlbauer, S. Uhlig, R. Bush, P. Francois, and O. Maennel. Evolution of Internet Address Space Deaggregation: Myths and Reality. *IEEE Journal on Selected Areas in Communications*, 28(8):1238–1249, October 2010.

[11] T. Bu, Gao L., and Towsley D. On characterizing BGP routing table growth. *Computer Networks*, 45(1):45–54, May 2004.

[12] A. Cabellos-Aparicio and D. Saucez. An architectural introduction to the Locator/ID Separation Protocol (LISP). Internet Draft (Work in Progress) draft-ietf-lisp-introduction-13, Internet Engineering Task Force, April 2015.

[13] IANA. Address Family Numbers. URL https://www.iana.org/assignments/address-family-numbers.

[14] D. Farinacci, D. Meyer, and J. Snijders. LISP Canonical Address Format (LCAF). RFC 8060, Internet Engineering Task Force, February 2017.

[15] F. Coras, J. Domingo-Pascual, D. Lewis, and A. Cabellos-Aparicio. An analytical model for loc/ID mappings caches. *IEEE/ACM Transactions on Networking*, 24 (1):506–516, February 2016.

[16] V. Fuller, D. Farinacci, D. Meyer, and D. Lewis. Locator/ID Separation Protocol Alternative Logical Topology (LISP+ALT). RFC 6836, Internet Engineering Task Force, January 2013.

[17] Y. Rekhter, S. Hares, and T. Li. A Border Gateway Protocol 4 (BGP-4). RFC 4271, Internet Engineering Task Force, January 2006.

[18] P. Mockapetris. Domain names - concepts and facilities. RFC 1034, Internet Engineering Task Force, November 1987.

[19] D. Lewis, D. Meyer, D. Farinacci, and V. Fuller. Interworking between Locator/ID Separation Protocol (LISP) and Non-LISP Sites. RFC 6832, Internet Engineering Task Force, January 2013.

[20] B. Carpenter and S. Brim. Middleboxes: Taxonomy and Issues. RFC 3234, Internet Engineering Task Force, February 2002.

[21] G. Papastergiou, G. Fairhurst, D. Ros, A. Brunstrom, K-J Grinnemo, P. Hurtig, N. Khademi, M. Tuxen, M. Welzl, D. Damjanovic, and S. Mangiante. Deossifying the internet transport layer: A survey and future perspectives. *IEEE Communications Surveys & Tutorials*, 19(1):619–639, Firstquarter 2017.

[22] S. Deering and R. Hinden. Internet Protocol, Version 6 (IPv6) Specification. RFC 8200, Internet Engineering Task Force, July 2017.

[23] D. Farinacci and B. Weis. Locator/ID Separation Protocol (LISP) Data-Plane Confidentiality. RFC 8061, Internet Engineering Task Force, February 2017.

[24] W. Ramirez, X. Masip-Bruin, M. Yannuzzi, R. Serral-Gracia, A. Martinez, and M.S. Siddiqui. A survey and taxonomy of ID/locator split architectures. *Computer Networks*, 60:13–33, February 2014.

[25] J. Kurose and K. Ross. *Computer networking: A top down approach sixth edition*. Pearson, 2012.

[26] E. Lear. NERD: A Not-so-novel Endpoint ID (EID) to Routing Locator (RLOC) Database. RFC 6837, Internet Engineering Task Force, January 2013.

[27] L. Jakab, A. Cabellos-Aparicio, F. Coras, D. Saucez, and O. Bonaventure. LISP-TREE: a DNS hierarchy to support the LISP mapping system. *IEEE Journal on Selected Areas in Communications*, 28(8):1332–1343, October 2010.

[28] J. Paillissé, A. Cabellos, V. Ermagan, and F. Maino. A Blockchain-based Mapping System. URL https://www.ietf.org/proceedings/98/slides/slides-98-lisp-lisp-a-blockchain-based-mapping-system-00.pdf.

[29] L. Mathy and L. Iannone. LISP-DHT: Towards a DHT to Map Identifiers onto Locators. In *Proceedings of the 2008 ACM CoNEXT Conference*, page 61. Association for Computing Machinery, December 2008.

[30] S. Brim, D. Farinacci, D. Meyer, and J. Curran. EID Mappings Multicast Across Cooperating Systems for LISP. Internet Draft (Work in Progress) draft-curran-lisp-emacs-00, Internet Engineering Task Force, November 2007.

[31] D. Farinacci and C. Cantrell. A Decent LISP Mapping System (LISP-Decent). Internet Draft (Work in Progress) draft-farinacci-lisp-decent-04, Internet Engineering Task Force, September 2019.

[32] S. Rose, M. Larson, D. Massey, R. Austein, and R. Arends. DNS Security Introduction and Requirements. RFC 4033, Internet Engineering Task Force, March 2005.

[33] F. Coras, D. Saucez, L. Iannone, and B. Donnet. On the performance of the LISP Beta Network. In *2014 IFIP Networking Conference*, pages 1–9. IEEE, June 2014.

[34] G. Louppe. INFO8002: Large-scale Data Systems, Lecture 9: Distributed Hash Tables. Université de Liège, 2018-2019.

[35] N. Bozic, G. Pujolle, and S. Secci. A tutorial on blockchain and applications to secure network control-planes. In *2016 3rd Smart Cloud Networks Systems (SCNS)*, pages 1–8. IEEE, December 2016.

[36] R. Moskowitz, D. Karrenberg, Y. Rekhter, E. Lear, and G.J. de Groot. Address allocation for private internets. RFC 1918, Internet Engineering Task Force, February 1996.

[37] D. Saucez, L. Iannone, and B. Donnet. A First Measurement Look at the Deployment and Evolution of the Locator/ID Separation Protocol. *SIGCOMM Computer Communication Review*, 43(2):37–43, April 2013.

[38] W. Diffie and M. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, November 1976.

[39] E. Rescorla and N. Modadugu. Datagram Transport Layer Security Version 1.2. RFC 6347, Internet Engineering Task Force, January 2012.

[40] D. Saucez, L. Iannone, and O. Bonaventure. Locator/ID Separation Protocol (LISP) Threat Analysis. RFC 7835, Internet Engineering Task Force, April 2016.

[41] P. Almasan, J. Paillisse, A. Rodriguez-Natal, P. Barlet-Ros, F. Coras, V. Ermagan, F. Maino, and A. Cabellos-Aparicio. Securing the control-plane channel and cache of pull-based ID/LOC protocols, 2018.

[42] C. Rossow. Amplification hell: Revisiting network protocols for DDoS abuse. In *Proceedings 2014 Network and Distributed System Security Symposium*, pages 1–15. Internet Society, February 2014.

[43] R. van Rijswijk-Deij, A. Sperotto, and A. Pras. DNSSEC and its potential for DDoS attacks. In *Proceedings of the 2014 Conference on Internet Measurement Conference*, pages 449–460. ACM Press, November 2014.

[44] G. Kambourakis, T. Moschos, D. Geneiatakis, and S. Gritzalis. Detecting DNS amplification attacks. In *Critical Information Infrastructures Security*, pages 185–196. Springer, 2008.

[45] Universitat Politècnica de Catalunya. LISPmon. URL `http://lispmon.net/`.

[46] T. Moraux. LISP, implémentation d'un looking glass et études préliminaires sur la mise en place d'un environnement de tests. Master's thesis, University of Liège, Belgium, 2014-2015.

[47] R. Doriguzzi-Corin, S. Millar, S. Scott-Hayward, J. Martinez-del Rincon, and D. Siracusa. LUCID: A practical, lightweight deep learning solution for DDoS attack detection. *IEEE Transactions on Network and Service Management*.

[48] H. Song, S. Dharmapurikar, J. Turner, and J. Lockwood. Fast hash table lookup using extended bloom filter. *ACM SIGCOMM Computer Communication Review*, 35(4):181–192, August 2005.

[49] G. Cormode and M. Hadjieleftheriou. Finding frequent items in data streams. *Proceedings of the VLDB Endowment*, 1(2):1530–1541, August 2008.

[50] P. Ferguson and D. Senie. Network Ingress Filtering: Defeating Denial of Service Attacks which employ IP Source Address Spoofing. RFC 2827, Internet Engineering Task Force, May 2000.

[51] R. Beverly, A. Berger, Y. Hyun, and k. claffy. Understanding the efficacy of deployed internet source address validation filtering. In *Proceedings of the 9th ACM SIGCOMM conference on Internet measurement conference*, pages 356–369. ACM Press, November 2009.

[52] D. Eastlake and M. Andrews. Domain Name System (DNS) Cookies. RFC 7873, Internet Engineering Task Force, May 2016.